On the π-calculus and distributed calculi for co-intuitionistic logic

Gianluigi Bellin

Dipartimento di Informatica, Università di Verona
Strada Le Grazie, 37134 Verona, Italy
gianluigi.bellin@univr.it

Summary. We reconsider early work on the π-calculus and linear logic ([6], written in 1992), asking whether *distributed* aspect of logical computation are expressed in the π-calculus translation. As a test case we propose a term assignment to Gentzen systems for linear co-intuitionistic logic (cfr [4]). We argue that this calculus has distinctive features of distributed computations and conclude that a translation such logics in some system of *distributed* π-calculus may better represent their computational properties than the 1992 version of the (synchronous) π-calculus.

1 Introduction

In the large body of literature about R. Milner's π-calculus, starting from [18], there has been a continuous stream of research on what Milner called the *synchronous* π-calculus; an early development of it is found in the translations of linear logic into the π-calculus, which were pursued both by S. Abramsky and by R. Milner around 1991.[1]

Linear logic [9] was regarded as a good candidate for representing *concurrent* logical computations: Girard himself had presented the representation of derivations in classical linear logic in the system of *proof nets* as a way to realize the "parallelization of the syntax"; in [9] this goal was regarded as achieved for the *multiplicative fragment* of linear logic. Therefore the encoding in the π-calculus of linear logic proofs, and in particular of proof-nets, and the representation of logical normalization of proofs through the transformation of the corresponding π-calculus processes were regarded as an interesting test both of π-calculus expressivity and of the claim of linear logic to be a logic for concurrency.

[1] We thank Neil Jones, Massimo Merro, Ugo Solitro and Carolyn Talcott for helpful conversations.

Documentation of that research is in the paper by Bellin and Scott [6][2]. The paper clearly indicates that one of Milner's motivation for development of that version of the synchronous calculus was the encoding of linear logic: as stated in the introduction of [6], this version of the π-calculus was "purposely supporting some of the logical rewriting envisioned by Abramsky". In particular, *contextual rewriting* is generally required by normalisation in logic and in the lambda calculus but becomes impossible within a term P in the scope of a *guarding prefix* $x(\tilde{z})Q$ or $\overline{x}\langle\tilde{a}\rangle Q$, namely, when no interaction involving subterms of Q is allowed to start until the channel x has acted. To minimize syntactic restrictions to interaction, the congruence $\omega_1\omega_2 P \equiv \omega_2\omega_1 P$ was introduced for arbitrary prefixes ω_1 and ω_3, when no free variable becomes bound and no bound variable becomes free. But Milner also allowed the use of the "guarding dot" $\omega.P$, regarded as an independent operator. The translation was successful for the *multiplicative fragment* **MLL** of linear logic, where the "guarding dot" was not used, but in the *multiplicative and additive* fragment the representability of cut-elimination was seriously limited by its use.

In the π-calculus there is an evident asymmetry between *receiving prefixes* $x(\tilde{z})Q$, that bind the variables $\tilde{z}$ in Q and *sending prefixes* $\overline{x}\langle\tilde{a}\rangle P$ which do not. In the case of *senders* there seems to be no reason to distinguish between $\overline{x}\langle\tilde{a}\rangle\overline{y}\langle\tilde{b}\rangle P$ and $\overline{y}\langle\tilde{b}\rangle\overline{x}\langle\tilde{a}\rangle P$ and even between $\overline{x}\langle\tilde{a}\rangle\overline{y}\langle\tilde{b}\rangle P$ and $\overline{y}\langle\tilde{b}\rangle\|\overline{x}\langle\tilde{a}\rangle\|P$, while *receivers* may impose an ordering, as in the case of $x(y)y(z)P$. This fact was exploited by Milner, who sought to allow non-guarding prefixes and at the same time to use name binding to simulate the effect of guarding prefixes. About ten years later Milner's goal was achieved: see, e.g., work by C. Laneve and B. Victor [15] (particularly the second encoding). Here guarding prefixes are simulated but the asymmetry between *senders* and *receivers* is removed and binding is performed only by the ν-operator (*abstraction*).

In Bellin and Scott [6] one also sees that research in linear logic was already interested in the notion of *polarity* and thus tried to *exploit* the asymmetry between senders and receivers rather than to eliminate it. The goal was to simulate the *information flow*, which occurs within proofs, in the π-calculus translation. The notion of an *information flow* within *intuitionistic* proofs (from *input formulas* A_I in the elimination part of proof-branches to *output formulas* C_O in the introduction parts) was well-known, but we learnt from Girard's proof-nets that a more complex information flow occurs within *classical* linear proofs as well.[3] Thus π-calculus translations were required to translate *input* and *output formulas* with an *receiving* and *sending* process, respectively.

[2] In 1991-2 the authors were in Edinburgh, learning the π-calculus from Robin Milner; also they were well aware of the work by Samson Abramsky, who gave a lecture on it in Edinburgh in 1992.

[3] This idea, developed at length in section 5 of [6], was also studied at the time by François Lamarche [14]. Later the "information flow" was related to Chu's construction and to the abstract form of game-semantics, see, e.g.. [2].

Girard's notion of a "parallelization of the syntax" through the representation of proofs as *multiple conclusions* proof-nets $\mathcal{R}_{A_1,\ldots,A_n}$ is based on the principle that all conclusions A_1, …, A_n must be regarded as equivalent "interaction ports": this is made possible by the facts that linear negation is an *involution*, (i.e., $A^{\perp\perp} = A$); thus, e.g., there is no reason to give "privileged access" to one of the two "ports" of an axiom $\overline{A\ A^{\perp}}$. The opposite assumption is made when a logic is regarded as *polarized.* Much work in linear logic in recent year has been based on the idea that polarized logics and polarized proof-nets offer the most fruitful results from the viewpoint of logical computation; certainly game semantics is naturally polarized and this fact carries much weight as we come to appreciate the rich results of the interaction between proof-theory and game-semantics. One could say that in their maturity linear logic and the π-calculus have proceeded in different directions: but this does not mean that research following earlier ideas and motivations must be sterile, only that it may need to exploit fresh new ideas.

Let's take the idea of a "parallel syntax" seriously. Consider the multiple conclusions $A_1, \ldots, A_n$ of a proof-net $\mathcal{R}$, representing a derivation in classical linear logic: can we regard such a derivation as a *concurrent and distributed process*? Suppose we could: certainly we would expect to recognize these features also in the π-calculus translation. Now consider the "ports" $x_1 : A_1, \ldots, x_n : A_n$ of a proof-net $\mathcal{R}$ in the π-calculus translation $\pi(\mathcal{R})$ and think of these "ports" as operated by distinct *agents* in distinct *locations.* In which sense the computation is *distributed* among them? A glance at the Abramsky-Milner translation of linear logic into the synchronous π-calculus translation in [6] shows that x_1, …, x_n are only channels through which an agent can access to some parts the process $\pi(\mathcal{R})$, other parts being accessible only through other ports. The computation may be seen as *concurrent*, in the sense that two mutually inaccessible processes compute in parallel, unless and until their scopes merge; but as a matter of fact the interaction between processes is tightly constrained through name-bindings (by the ν-operator and receiving prefixes) and it is hard to see it as the action of independent agents in a distributed context.

This fact is not surprising. Certainly the π-calculus is a calculus of *concurrent interaction and communication* between processes; but is it also a *distributed* calculus? Since the 1990s *distributed versions* of the π-calculus have been provided (see, e.g., [1, 19, 12, 11] and the book [10]), as a theoretical framework for the study of real distributed systems in presence of nodes and link failure, or for the solution of concrete problems, such as safety and control of mobile code. Distributed π-calculi extend the π-calculus with symbols and actions for *locations* and sometimes differ in the treatment of basic operators: in some early papers, e.g., in [1, 19], communication is global, while for others [12, 11] only processes in the same location can interact. Some version of the distributed π-calculus would seem appropriate to highlight distributed features of logical computations in logics with "parallel syntax".

1.1 A test case: co-intuitionistic linear logic.

In this paper we propose a term assignment to *co-intuitionistic linear logic*, in a calculus which appears to be “more distributed” than the 1992 version of the synchronous π-calculus. We may think of co-intuitionistic logic as being about *making hypotheses*. Its consequence relation has the form

$$H \vdash H_1, \ldots, H_n. \tag{1}$$

Suppose H is a hypothesis: which (disjunctive sequence of) hypotheses H_1 *or* … *or* H_n can we prove to be compatible with 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. Here we insist on interpreting the consequence relation as *distributed*, i.e., we think of the alternatives H_1, …, H_n in (1) as lying in different *locations*.

The main connectives are *subtraction* $A \smallsetminus B$ (*possibly* A *and not* B) and Girard’s *par* $A \wp B$. Natural Deduction inference rules for *subtraction* are as follows. Here again we insist in the *distributed interpretation* of multiple conclusion sequents.

$$\smallsetminus\text{-}intro\ \frac{H \vdash \Gamma, C \qquad D \vdash \Delta}{H \vdash \Gamma, C \smallsetminus D, \Delta} \qquad \smallsetminus\text{-}elim\ \frac{H \vdash \Delta, C \smallsetminus D \qquad C \vdash D, \Upsilon}{H \vdash \bullet, \Delta, \Upsilon}$$

Notice that in the $\smallsetminus$-*elimination* rule the evidence of possible compatibility between C and D given by the *right premise* has become *inconsistent with the hypothesis* $C \smallsetminus D$ in the left premise, thus in the conclusion we drop D and add $\bullet$ to remember the *removed evidence* of this now inconsistent alternative.

However if the *left premise* of $\smallsetminus$-*elim.*, proving that $C \smallsetminus D$ with Δ is compatible with H, has been obtained by a $\smallsetminus$-*introduction*, this inference has the form

$$\frac{H \vdash \Delta_1, C \qquad D \vdash \Delta_2}{H \vdash \Delta_1, \Delta_2, C \smallsetminus D}.$$

But then the pair of *introduction/elimination* rules can be eliminated: using the *removed evidence* that D with Υ are compatible with C (*right premise* of $\smallsetminus$-*elim.*) we can conclude that $\Delta_1, \Delta_2, \Upsilon$ are compatible with H. This is, in a nutshell, the principle of normalization (or *cut-elimination*) for subtraction.

We provide a term assignment to co-intuitionistic linear logic with the property that distinct terms t_1, … t_n are assigned to the formulas H_1, …, H_n, respectively, in the given context (1). A term t_i formally expresses a *parcel of evidence* that H and H_i may be compatible in the given context; thus the terms t_i are *distributed* in the locations of the formulas H_i.[4]

[4] This feature is also found in the term assignment to *full intuitionistic linear logic* [13], although in most calculi, including Crolard’s [8] for subtraction, terms are assigned to a selected formula in the succedent.

The calculus presented here is the linear version of the term assignment to full co-intuitionistic logic in [3, 4]. Since it is designed to decorate precisely the derivations that are *dual* of intuitionistic linear derivations, it may be called *dual linear lambda calculus*. The calculus is not based on *name passing*; the operators `make-coroutine` (`mkc`) and `postpone` (`postp`) associated with *subtraction introduction* and *elimination* are similar to **CPS** terms (see [20] and also [17]); in fact, we may regard our calculus as consisting of *intuitionistic continuations*. A distinctive feature is the absence of a ν-operator, as the dependence between binding constructs is dealt with by replacing variables with functional terms reminiscent of *Herbrand functions*. The *distributed* character of the dual linear calculus is evident not only in the locations of terms, buts also in other features, such as the "broadcasting" of *remote substitutions* (see in particular Remark 4 in Section 6).

2 Basic Synchronous π-calculus

We review here the *basic synchronous π-calculus* presented in [6].

We are given a countable family $\mathcal{X}$ of variables (*names*), denoted by $a, d, c, \ldots, x, y, z$; we denote vectors of names with $\tilde{x}$, $\tilde{y}$, etc.

Definition 1. (**Grammar**) *The π-calculus* processes P *are denoted by the expressions* (π-terms) *defined by the following grammar:*

$$P, Q \;:=\; \bullet \mid (\nu\tilde{x})P \mid (P \| Q) \mid \overline{x}\langle\tilde{y}\rangle P \mid x(\tilde{y})P$$

Here

(o) $\bullet$ *denotes the null process.* $(P\|Q)$ *denotes the process resulting from* P *and* Q *acting concurrently.*
(i) *A* prefix *of the form* $\overline{x}\langle\tilde{y}\rangle$ (sender) *or* $x(\tilde{y})$ (receiver) *denotes a* channel *named* x *through which the names in* $\tilde{y}$ *can be sent or received, respectively.*
(ii) *In the π-terms* $x(\tilde{y})P$ *and* $(\nu\tilde{x})P$ *the receiver* $x(\tilde{y})$ *and the expression* $(\nu\tilde{y})$ (hiding) *are* name-binding operators *and* P *is their* scope.

Prefixes are denoted by π *and* ω *denotes either a prefix or a hiding operator.*

(**Free and bound names**) *The sets* $fn(P)$ *and* $bn(P)$ *of* free *and* bound *names in* P *are defined as follows:*

(a) $fn(\bullet) = bn(\bullet) = \emptyset$;
(b) $fn(P\|Q) = fn(P) \cup fn(Q)$; $bn(P\|Q) = bn(P) \cup bn(Q)$;
(c) $fn(\overline{x}\langle\tilde{y}\rangle P) = \{x\} \cup \tilde{y} \cup fn(P)$; $bn(\overline{x}\langle\tilde{y}\rangle P) = bn(P)$;
(d) $fn(x(\tilde{y})P) = fn(P) \setminus \tilde{y} \cup \{x\}$; $bn(x(\tilde{y})P) = bn(P) \cup \tilde{y}$;
(e) $fn((\nu\tilde{y})P) = fn(P) \setminus \tilde{y}$; $bn((\nu\tilde{y})P) = bn(P) \cup \tilde{y}$.

We define α-equivalence, renaming of bound variables and capture-avoiding simultaneous substitution $P[\tilde{y}/\tilde{x}]$ *as usual.*

(**Congruence**) *A* congruence relation $\equiv$ *on π-terms is defined as follows:*

1. *$\omega_1\omega_2 P \equiv \omega_2\omega_1 P$ provided no free variable becomes bound and no bound variable becomes free.*
2. *$\omega(P\|Q) \equiv \omega P\|Q$, provided $bn(\omega) \cap fn(Q) = \emptyset$.*
3. *Parallel composition $\|$, regarded as a binary operator on processes, satisfies the axioms of a commutative monoid with unit $\bullet$.*
4. *$(\nu x)\bullet \ \equiv \ \bullet$.*

(**Basic 1-step reduction**)

$$(\overline{x}\langle\tilde{y}\rangle P \ \| \ x(\tilde{z})Q \quad \succ_1 \quad (P\|Q[\tilde{y}/\tilde{z}])$$

Rewriting is contextual and modulo $\equiv$, *so that for all context $\mathcal{C}$, if $P' \equiv P$ and $Q \equiv Q'$, then we have $P \succ Q \Rightarrow \mathcal{C}[P'] \succ_1 \mathcal{C}[Q']$.*

Remark 1. It is important to notice that in this version of the π-calculus the notion of the *scope* of a prefix is essentially related to binding. For instance, since $\overline{x}\langle\tilde{y}\rangle(P\|Q) \ \equiv \ (\overline{x}\langle\tilde{y}\rangle P\|Q)$ we have $\overline{x}\tilde{y}x(\tilde{z}P \ \equiv \ (\bullet\|x(\tilde{z}P) \succ_1 P[\tilde{y}/\tilde{z}]$. To constrain the reduction process in the synchronous calculus one uses the *guarding* operator (*dot*): $\pi.P$.

Remark 2. We cannot survey here the recent literature in which ideas related to the *synchronous π-calculus* have been developed, but the paper by Laneve and Victor [15] is particularly relevant, as it shows how *guarding prefixes* can be encoded in a calculus where the only binding operator is abstraction (the ν-operator).

3 Multiplicative co-Intuitionistic Logic

The logic we apply the π-calculus translation to is **co-IMLL**, the *dual* of Multiplicative Intuitionistic Linear Logic (**IMLL**). Intuitionistic Linear Logic $\mathbf{IMLL}^{\multimap\otimes}$ with *linear implication* ($\multimap$) and *tensor product* ($\otimes$) is formalized in Gentzen calculi with single-conclusion sequents

$$x_1 : A_1, \ldots x_n : A_n \vdash t : A$$

which can be decorated with linear lambda terms with tensor. The dual logic $\mathbf{co\text{-}IMLL}^{\setminus\wp}$ on the connectives *subtraction* ($A \setminus B$) and *par* ($A\wp B$) is formalized in a sequent calculus $\mathbf{MLJ}^{\setminus\wp}$ with *single-premise* and *multiple-conclusions* sequents

$$x : C \vdash t_1 : C_1, \ldots, t_n : C_n.$$

Here the terms t_1, ..., t_n belong to the *dual linear lambda calculus*, or simply *dual linear calculus* defined in Section 5.

Given a countable sequence of elementary formulas denoted by $\eta_1, \eta_2, \ldots$ the language of $\mathbf{co\text{-}IMLL}^{\setminus\wp}$ is given by the following grammar:

$$A, B \;:=\; \eta \mid A \smallsetminus B \mid A \wp B$$

Here $\eta = \mathcal{H}p$ expresses the hypothesis that proposition p is true. If A and B are hypothetical expressions, then the intended meaning of $A \smallsetminus B$ is *possibly A and not B*; we take an intuitive explanation of $A \wp B$ (A *par* B) to be *it is not possible that definitely not A and definitely not B*. For co-intutionistic logic as a logic of hypotheses, see [4].

Sequent calculus $\mathbf{MLJ}^{\smallsetminus\wp}$

The rules of the sequent calculus $\mathbf{MLJ}^{\smallsetminus\wp}$ for Multiplicative co-Intuitionistic Linear Logic with *subtraction* and *par* are given in Table 1.

axiom $$A \vdash A$$	*cut* $$\frac{E \vdash \Gamma, A \qquad A \vdash \Delta}{E \vdash \Gamma, \Delta}$$
$\smallsetminus$-R $$\frac{E \vdash \Gamma, C \qquad D \vdash \Delta}{E \vdash \Gamma, C \smallsetminus D, \Delta}$$	$\smallsetminus$-L $$\frac{C \vdash D, \Delta}{C \smallsetminus D \vdash \Delta}$$
$\wp$-R $$\frac{E \vdash \Gamma, C_0, C_1}{E \vdash \Gamma, C_0 \wp C_1}$$	$\wp$-L $$\frac{C_0 \vdash \Gamma_0 \qquad C_1 \vdash \Gamma_1}{C_0 \wp C_1 \vdash \Gamma_0, \Gamma_1}$$

Table 1. Sequent Calculus $\mathbf{MLJ}^{\smallsetminus\wp}$

3.1 Example.

Consider the Petri net **N** in Figure 1 and the computation resulting from the given initial marking, namely, using resources A and B to fire the first transition and obtain C and then using a resource D to fire the second transition and obtain E.
Such computation can be represented in Intuitionistic Multiplicative Linear Logic **IMLL** by a (cut free) sequent derivation of

$$\underbrace{(A, A, B, D, D}_{M_1}, \underbrace{A \multimap (B \multimap C)}_{T_1}), \underbrace{C \multimap (D \multimap E)}_{T_2}) \vdash \underbrace{E \otimes (A \otimes D)}_{M_2} \tag{2}$$

as shown, e.g., in [16].

The same computation can be encoded in an easy **CCS** computation:

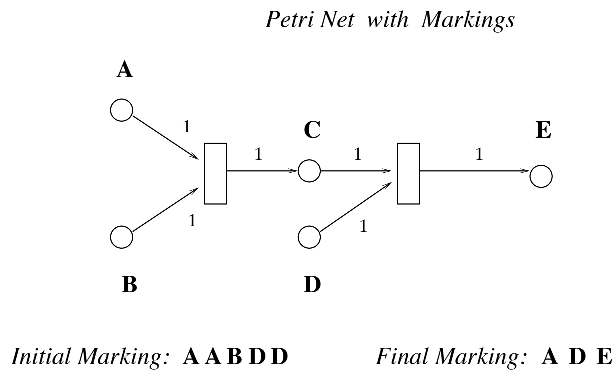


Fig. 1. A Petri Net.

$$a\|a\|b\|d\|d\|(\nu xy)\big(\overline{a}.x\|b.\overline{x}.y\|d.\overline{y}.e\big) \quad \succ \quad a\|d\|e \tag{3}$$

But here the use of *guarding prefixes* of **CCS** (or some encoding of them in the π-calculus) is essential, so there is no way to represent the cut-elimination process of **IMLL** in **CCS**: for this purpose, as pointed out above, *contextual rewriting* would be required *within guarded processes*.

If we decorate **IMLL** natural deduction or sequent derivations with the *linear λ-terms* with $\otimes$ and `let` constructs as, e.g., in [7], then we have a full and faithful representation of normalization or cut-elimination in **IMLL** and this allows us to represent *other computation strategies*. For instance, in our simple example we may load the nodes B, D and A in this order and still the computation would go through. Thus we have derivations $\mathcal{D}_{11}$, $\mathcal{D}_{12}$, $\mathcal{D}_{21}$ and $\mathcal{D}_{22}$ decorated with the following terms:

$$\mathcal{D}_{11}:$$
$$g : T_2, f : T_1 \vdash \underbrace{\lambda b \lambda d \lambda a.g(fab)d}_{u} : B \multimap (D \multimap (A \multimap E))$$

$$\mathcal{D}_{12}:$$
$$y_1 : A \otimes A, b : B, y_2 : D \otimes D, h : B \multimap (D \multimap (A \multimap E)) \vdash \ell : E \otimes (A \otimes D)$$

where $\ell = \texttt{let}\ y_2\ \texttt{be}\ d \otimes d'\ \texttt{in}\ \ell_1$ and $\ell_1 = \texttt{let}\ y_1\ \texttt{be}\ a \otimes a'\ \texttt{in}\ (hbda) \otimes (a' \otimes d')$

$$\mathcal{D}_{21}: \qquad\qquad \mathcal{D}_{22}:$$
$$a : A, a' : A \vdash a \otimes a' : A \otimes A \qquad d : D, d' : D \vdash d \otimes d' : D \otimes D$$

and we obtain a derivation $\mathcal{D}_0$ *with cut* as follows:

$$
\dfrac{\mathcal{D}_{21} \qquad \dfrac{\mathcal{D}_{22} \qquad \dfrac{\mathcal{D}_{11} \qquad\qquad \mathcal{D}_{12}}{y_1 : A \otimes A, b : B, y_2 : D \otimes D, g : T_2, f : T_1 \vdash \ell[u/h] : E \otimes (A \otimes D)}\,cut_1}{d : D, d' : D, y_1 : A \otimes A, b : B, g : T_2, f : T_1 \vdash \ell[u/h][d \otimes d/y_2] : E \otimes (A \otimes D)}\,cut_2}{d : D, d' : D, a : A, a' : A, b : B, g : T_2, f : T_1 \vdash \ell[u/h][d \otimes d'/y_2][a \otimes a'/y_1] : E \otimes (A \otimes D)}\,cut_3
$$

As expected, by eliminating cuts we get

$$
d : D, d' : D, b : B, a : A, a' : A, g : T_2, f : T_1 \vdash g(fab)d : E
$$

3.2 The example in co-Intuitionistic Multiplicative Linear Logic

Here we consider a dual representation of the same Petri Net computation as derivations of the sequent R :

$$
\underbrace{E\wp(A\wp D)}_{M_2^{\perp}} \vdash \underbrace{(E \setminus D) \setminus C}_{T_2^{\perp}}, \underbrace{(C \setminus B) \setminus A}_{T_1^{\perp}}, \underbrace{A, A, B, D, D}_{M_1^{\perp}}
$$

The following is a cut free derivation $\mathcal{D}_R$ of R in $\mathbf{MLJ}^{\setminus\curlyvee}$:

$$
\dfrac{\dfrac{\dfrac{E \vdash E \qquad D \vdash D}{E \vdash E \setminus D, D}\setminus\text{-R} \qquad \dfrac{\dfrac{\dfrac{C \vdash C \qquad B \vdash B}{C \vdash C \setminus B, B}\setminus\text{-R} \qquad A \vdash A}{C \vdash (C \setminus B) \setminus A, A, B}\setminus\text{-R}}{}}{E \vdash (E \setminus D) \setminus C, T_1, A, B, D}\setminus\text{-R} \qquad \dfrac{A \vdash A \qquad D \vdash D}{A\wp D \vdash A, D}\wp\text{-L}}{E\wp(A\wp D) \vdash T_1, T_2, A, A, B, D, D}\setminus\text{-R}
$$

This derivation represents the Petri net computation showing the reachability of E in Figure 1 in the linear co-intuitionistic sequent calculus $\mathbf{MLL}^{\setminus\wp}$.

4 Abramsky's π-calculus processes for $\mathbf{MLJ}^{\setminus\wp}$

The following encoding is the same as that of classical multiplicative linear logic in [6]. Notice however that here we exploit the "logical flow" of co-intuitionistic logic by decorating the formula in the antecedent with a *receiver* and the formulas in the succedent with a *sender*.

Logical rule	**π-translation**
$I_{xy} \; : \; x : A \vdash \overline{y} : A$	$I_{xy} = x(a)\overline{y}\langle a \rangle$

cut

$$
\dfrac{\begin{matrix}\vdots \\ F_{v\tilde{x}x} : \\ v : D \vdash \tilde{x} : \Gamma, x : A\end{matrix} \qquad \begin{matrix}\vdots \\ G_{y\tilde{y}} : \\ y : A \vdash \tilde{y} : \Delta\end{matrix}}{\mathbf{Cut}^z(F, G)_{\tilde{x}\tilde{y}} : \quad v : D \vdash \tilde{x} : \Gamma, \tilde{y} : \Delta}
\qquad
\begin{matrix}\mathbf{Cut}^z(F, G)_{\tilde{x}\tilde{y}} \quad = \\ = \nu z\Big(F_{v\tilde{x}}[z/x] \parallel G_{\tilde{y}}[z/y]\Big)\end{matrix}
$$

subtraction introduction

$$\frac{\begin{array}{cc} \vdots & \vdots \\ F_{v\tilde{x}x}: & G_{y\tilde{y}}: \\ v:D\vdash \tilde{x}:\Gamma, x:A & y:B\vdash \tilde{y}:\Delta \end{array}}{\mathbf{SI}^{xy}_z(F,G)_{\tilde{x}\tilde{y}}:\quad v:D\vdash \tilde{x}:\Gamma, z:A\setminus B, \tilde{y}:\Delta} \qquad \begin{array}{l}\mathbf{SI}^{xy}_z(F,G)_{\tilde{x}\tilde{y}} \quad = \\ = \nu xy\Big(\overline{z}\langle xy\rangle\ (F_{v\tilde{x}x} \parallel G_{y\tilde{y}})\Big)\end{array}$$

subtraction elimination

$$\frac{\begin{array}{c}\vdots \\ F_{xy\tilde{x}}:\quad x:A\ \vdash\ y:B, \tilde{x}:\Gamma\end{array}}{\mathbf{SE}^{xy}_z(F)_{\tilde{x}}:\quad z:A\setminus B\ \vdash\ \tilde{x}:\Gamma} \qquad \mathbf{SE}^{xy}_z(F)_{\tilde{x}}\ =\ z(xy)F_{xy\tilde{x}}$$

4.1 The Cut Algebra for $\mathbf{MLJ}^{\setminus}$

Symmetric Reductions:

$$\begin{array}{lrl}
(1) & \mathbf{Cut}^z(F_{\tilde{x}}, G_{\tilde{y}}) & = \mathbf{Cut}^z(G_{\tilde{y}}, F_{\tilde{x}}) \\
(2) & \mathbf{Cut}^x(F_{\tilde{v}x}, I_{xy}) & \succ F_{\tilde{v}x}[y/x] \\
(3) & \mathbf{Cut}^z(\mathbf{SI}^{xy}_z(F_x, G_y), \mathbf{SE}^{xy}_z(H_{xy}))_{\tilde{x}\tilde{y}\tilde{w}} & \succ \mathbf{Cut}^y(\mathbf{Cut}^x(F_x, H_{xy}), G_y)_{\tilde{x}\tilde{y}\tilde{w}} \\
(4) & & \equiv \mathbf{Cut}^y(G_y, \mathbf{Cut}^x(F_x, H_{xy}))_{\tilde{x}\tilde{y}\tilde{w}}
\end{array}$$

Commutative Reductions: If $\mathbf{SE}^{cd}_v$ and $\mathbf{SI}^{cd}_v$ do not react with x [5] and neither c nor d occurs in H, then

$$\begin{array}{ll}
(5) & \mathbf{Cut}^x(\mathbf{SE}^{cd}_v(F_{cdx}), H_x)_{\tilde{x}\tilde{w}} = \mathbf{SE}^{cd}_v(\mathbf{Cut}^x(F_{cdx}, H_x))_{\tilde{x}\tilde{w}} \\
(6) & \mathbf{Cut}^x(\mathbf{SI}^{cd}_v(F_c, G_{dx}), H_x)_{\tilde{x}\tilde{y}\tilde{w}} = \mathbf{SI}^{cd}_v(F_c, \mathbf{Cut}^x(G_{dx}, H_x))_{\tilde{x}\tilde{y}\tilde{w}} \text{ and} \\
 & \mathbf{Cut}^x(\mathbf{SI}^{cd}_v(F_{cx}, G_{dx}), H_x)_{\tilde{x}\tilde{y}\tilde{w}} = \mathbf{SI}^{cd}_v(\mathbf{Cut}^x(F_{cx}, H_x), G_d)_{\tilde{x}\tilde{y}\tilde{w}}.
\end{array}$$

Theorem 1. (**Soundness**) *Let $\mathcal{D}$ be a proof in* $\mathbf{MLJ}^{\setminus}$ *and let $\pi(\mathcal{D})$ be its π-calculus translation.*

(i) If $\mathcal{D} \succ'_{\mathcal{D}}$ by a 1-step symmetric reduction in the cut-elimination process, then $\pi(\mathcal{D}) \succ_1 \pi(\mathcal{D}')$ in the synchronous π-calculus.
(ii) If $\mathcal{D} \succ'_{\mathcal{D}}$ by a 1-step commutative reduction in the cut-elimination process, then $\pi(\mathcal{D}) \equiv \pi(\mathcal{D}')$ in the synchronous π-calculus.

(see [6]).

4.2 π calculus processes and cut algebra for $\mathbf{MLJ}^{\setminus\curlyvee}$

To the logical rules and π calculus processes for $\mathbf{MLJ}^{\setminus}$ we add the following rules and translations:

[5] In the context of this translation it suffice to assume that $c \neq x \neq d$.

Logical rule	**π-translation**
disjunction introduction $$\dfrac{\begin{matrix}\vdots\\ F_{xy\tilde{x}}: \quad v:D \vdash \overline{x}:A, \overline{y}:B, \tilde{x}:\Gamma\end{matrix}}{\mathbf{DI}_z^{xy}(F)_{\tilde{x}}: \quad v:D \vdash z:A \curlyvee B, \tilde{x}:\Gamma}$$	$\mathbf{DI}_z^{xy}(F)_{\tilde{x}} \;=\; \nu xy\ (\overline{z}\langle xy\rangle F_{xy\tilde{x}})$
disjunction elimination $$\dfrac{\begin{matrix}\vdots\\ G_{x\tilde{x}}: \quad x:A \vdash \tilde{x}:\Gamma\end{matrix} \qquad \begin{matrix}\vdots\\ H_{y\tilde{y}}: \quad y:B \vdash \tilde{y}:\Delta\end{matrix}}{\mathbf{DE}_z^{xy}(G,H)_{\tilde{x}\tilde{y}}: \quad z:A \curlyvee B \vdash \tilde{x}:\Gamma, \tilde{y}:\Delta}$$	$\mathbf{DE}_z^{xy}(G,H)_{\tilde{x}\tilde{y}} = \Big(z(xy)\ (G_{x\tilde{x}} \parallel H_{y\tilde{y}})\Big)$

To the *Symmetric Reductions* we add the following cases:

(3') $\mathbf{Cut}^z(\mathbf{DI}_z^{xy}(F_{xy}), \mathbf{DE}_z^{xy}(G_x, H_y))_{\tilde{x}\tilde{y}\tilde{w}} \succ \mathbf{Cut}^y(\mathbf{Cut}^x(F_{xy}, G_x), H_y)_{\tilde{x}\tilde{y}\tilde{w}}$
(4') $\equiv \mathbf{Cut}^x(\mathbf{Cut}^y(F_{xy}, H_y), G_x)_{\tilde{x}\tilde{y}\tilde{w}}$

To the *Commutative Reductions* we add the following cases:
If $\mathbf{DI}_v^{cd}$, $\mathbf{DE}_x^{cd}$ do not react with x and neither c nor d occurs in H, then

(5') $\mathbf{Cut}^x(\mathbf{DI}_v^{cd}(F_{cdx}), H_x)_{\tilde{x}\tilde{w}} = \mathbf{DI}_v^{cd}(\mathbf{Cut}^x(F_{cdx}, H_x))_{\tilde{x}\tilde{w}}$
(6') $\mathbf{Cut}^x(\mathbf{DE}_v^{cd}(F_c, G_{dx}), H_x)_{\tilde{x}\tilde{y}\tilde{w}} = \mathbf{DE}_v^{cd}(F_c, \mathbf{Cut}^x(G_{dx}, H_x))_{\tilde{x}\tilde{y}\tilde{w}}$ and
$\mathbf{Cut}^x(\mathbf{DE}_v^{cd}(F_{cx}, G_{dx}), H_x)_{\tilde{x}\tilde{y}\tilde{w}} = \mathbf{DE}_v^{cd}(\mathbf{Cut}^x(F_{cx}, H_x), G_d)_{\tilde{x}\tilde{y}\tilde{w}}$.

With these definitions the *Soundness Theorem* still holds for $\mathbf{MLJ}^{\prec\wp}$.

An example of π-calculus translation and computation is given in the Appendix (Section 8).

4.3 Proof-theoretic refinements

In [6], sections 4 and 5, representations of proofs in Classical Multiplicative Linear Logic (**MLL**) are considered for which not only *Soundess* (Theorem 1) but also *Local Fullness* holds of Abramsky's translation.

Given a logic **L***, for suitable representations of proofs and of proof-normalization in* **L** *and for their Abramsky translation* π *into* π*-calculus terms,* **local Fullness** *is the following property:*

Let $\mathcal{R}$ *be a proof in* **L** *and suppose* $\pi(\mathcal{R}) \succ Q$*. Then in* **L** *there exists a proof* $\mathcal{R}'$ *such that* $\pi(\mathcal{R}') = Q$ *and* $\mathcal{R} \succ \mathcal{R}'$*.*

For the sequent calculus $\mathbf{MLJ}^{\prec\wp}$ for Multiplicative co-Intuitionistic Logic, as well as for the sequent calculus for Classical Multiplicative Linear Logic **MLL**, Local Fullness holds only *modulo permissible permutations of inferences* (e.g., the commutative reductions above). On the other hand the representation of **MLL** proofs as *proof nets* [9] identifies sequent calculus proofs *modulo permutations of inferences* and therefore is suitable for a Local Fullness result.

It would be possible to follow Section 5 of [6] and develop a theory of *proof-nets with orientations* for *linear co-intuitionistic logic* simply by *dualizing the input-output orientations*; similarly, one would obtain Lamarche's *essential nets* for *linear co-intuitionistic logic* [14]. For instance, simply by reversing the roles of *input/output* we obtain the following translations:

$$(A \smallsetminus B)_O =_{df} A_O \otimes B_I \qquad (A \smallsetminus B)_I =_{df} A_I \wp B_O$$
$$(A \wp B)_O =_{df} A_O \wp B_O \qquad (A \wp B)_I =_{df} A_I \otimes B_I$$

We cannot pursue the topic here.

5 A dual linear calculus for co-IMLL$^{\smallsetminus\wp}$

We present the grammar and the basic definitions of our dual linear calculus for the fragment of linear co-intuitionistic logic with subtraction and disjunction.

Definition 2. *We are given a countable set of* free variables *(denoted by x, y, z …), and a countable set of* unary functions *(denoted by* $\mathtt{x}, \mathtt{y}, \mathtt{z}, \ldots$*).*

(i) *Terms are defined by the following grammar:*

$$t, u \ := x \mid \mathtt{x}(t) \mid t \wp u \mid \mathtt{casel}(t) \mid \mathtt{caser}(t) \mid \mathtt{mkc}(t, \mathtt{x})$$

(ii) *Let t_1, t_2, … an enumeration in a given order of all the* terms *freely generated by the above grammar starting with a special symbol $*$ and no variables (a selected variable a would also do the job). Thus we have a fixed* bijection *$t_i \mapsto x_i$ between terms and free variables.*

(iii) *Moreover, if t is a term and u is a term such that y occurs in u, then* $\mathtt{postp}(\mathtt{y} \mapsto u\{y := \mathtt{y}(t)\}, t)$ *is a p-term.*

We use the abbreviations $(t \to \mathtt{y})$ *for* $\mathtt{mkc}(t, \mathtt{y})$ *and* $\xleftarrow{\mathtt{z} \mapsto u} w$ *for* $\mathtt{postp}(\mathtt{z} \mapsto u, w)$.

Notice that a p-term cannot be a subterm of other terms.

Definition 3. (i) *The* free variables *$FV(\ell)$ in a term are defined as follows:*

$$\begin{aligned} FV(x) &= \{x\} \\ FV(\mathtt{x}(t) &= FV(t) \\ FV(t \wp u) &= FVt \cup FVu \\ FV(\mathtt{casel}(t)) = FV(\mathtt{caser}(t)) &= FV(t) \\ FV(\mathtt{mkc}\ (t, \mathtt{x}) &= FV(t) \\ FV(\mathtt{postp}(\mathtt{x} \mapsto u, t) &= FV(u) \cup FV(t). \end{aligned}$$

(ii) *A* computational context *$\mathcal{S}_x$ is a set of terms and p-terms containing the free variable x and no other free variable. We may represent a computational context as a list κ of terms and p-terms.*

Definition 4. *Substitution of a term t for a free variable x in a in a list of terms κ is defined as follows:*

$$
\begin{array}{rlrl}
x\{x:=t\} = t, & & y\{x:=t\} = y \text{ if } x \neq y; \\
\mathtt{y}(u)\{x:=t\} &= \mathtt{y}(u\{x:=t\}); & (r\wp u)\{x:=t\} = r\{x:=t\}\wp u\{x:=t\}) \\
\mathtt{casel}(r)\{x:=t\} = \mathtt{casel}(r\{x:=t\}), & & \mathtt{caser}(r)\{x:=t\} = \mathtt{caser}(r\{x:=t\}); \\
\mathtt{mkc}(r,\mathtt{y})\{x:=t\} &= \mathtt{mkc}(r\{x:=t\},\mathtt{y}), \\
\mathtt{postp}(\mathtt{y}\mapsto(u),s)\{x:=t\} &= \mathtt{postp}(\mathtt{y}\mapsto(u\{x:=t\}),s\{x:=t\}). \\
()\{x:=t\} = () & & (u\cdot\kappa)\{x:=t\} = t\{x:=t\}\cdot\kappa\{x:=t\}
\end{array}
$$

Definition 5. *β-reduction of a* redex $\mathcal{R}ed$ *in a* computational context $\mathcal{S}_x$ *is defined as follows.*
(i) *If $\mathcal{R}ed$ is a term u of the following form, then the reduction is local and consists of the rewriting $u \rightsquigarrow_\beta u'$ in $\mathcal{S}_x$ as follows:*

$$\mathtt{casel}\ (t\wp u) \rightsquigarrow_\beta\ t; \qquad \mathtt{caser}\ (t\wp u) \rightsquigarrow_\beta\ u.$$

(ii) *If $\mathcal{R}ed$ has the form $\xleftarrow{\mathtt{z}\mapsto u} (t \to \mathtt{y})$, i.e., $\mathtt{postp}(\mathtt{z}\mapsto u, \mathtt{mkc}(t,\mathtt{y}))$, then $\mathcal{S}_x$ has the form*

$$\mathcal{S}_x \;=\; \mathcal{R}ed, \quad \overline{\kappa}, \quad \overline{\zeta}_{\mathtt{y}}, \quad \overline{\xi}_{\mathtt{z}}$$

where $\mathtt{y}(t)$ occurs in $\overline{\zeta}_{\mathtt{y}}$ and $\mathtt{z}((t \to \mathtt{y}))$ occurs in $\overline{\xi}_{\mathtt{z}}$ and neither $\mathtt{y}(t)$ nor $\mathtt{z}((t \to \mathtt{y}))$ occurs in $\overline{\kappa}$. Using our bijection between the set of terms and the free variables, we can write $y = \mathtt{y}(t)$ and $z = \mathtt{z}((t \to \mathtt{y}))$; then a reduction *of $\mathcal{R}ed$ transforms the computational context as follows:*

$$\mathcal{S}_x \quad \rightsquigarrow \quad \overline{\kappa}, \quad \overline{\zeta}\{y := u\{z := t\}\}, \quad \overline{\xi}\{z := t\}. \tag{4}$$

Thus for $\overline{\zeta} = u_1, \ldots, u_k$ and $\overline{\xi} = r_1, \ldots, r_m$ we have:

$$
\begin{array}{rcl}
\overline{\xi}\{z := t\} &=& r_1\{z := t\}, \ldots, r_m\{z := t\}; \\
\overline{\zeta}\{y := u\{z := t\}\} &=& u_1\{y := u\{z := t\}\}, \ldots, u_k\{y := u\{z := t\}\}.
\end{array}
$$

6 Term assignment to $\mathbf{MLJ}^{\smallsetminus\wp}$

Definition 6. (**term assignment**) *The assignment of terms of the dual linear calculus to sequent calculus derivation in* $\mathbf{MLJ}^{\smallsetminus\wp}$ *is given in Table 2.*

Remark 3. (*Bound variables and α-conversion.*) Since in our calculus the binding of a free variable x is expressed by the substitution of x with a term $\mathtt{x}(t)$, how can we avoid that the same term $\mathtt{x}(t)$ may come to represent distinct bindings of the variable x? Since in the case of *linear co-intuitionistic logic* no copying of subtrees is produced by the normalization procedure, here such an undesirable effect may only arise if *two distinct occurrences* of the

Labelled Sequent Calculus $\mathbf{MLJ}^{\smallsetminus \wp}$
identity rules
logical axiom:
$x : C \;\vdash\; \pi : \bullet \mid\; x : C$
cut:
$$\frac{x : E \;\vdash\; \pi_1 : \bullet \mid \overline{\ell} : \Upsilon_1\ ,\ t : C \qquad y : C \;\Rightarrow\; \pi_2 : \bullet \mid \overline{\kappa} : \Upsilon_2}{x : E \;\vdash\; \pi_1, \pi_2\{y := t\} : \bullet \mid \overline{\ell} : \Upsilon_1, \overline{\kappa}\{y := t\} : \Upsilon_2}$$
logical rules
$\smallsetminus$ *right:*
$$\frac{x : E \;\vdash\; \pi_1 : \bullet \mid \overline{\ell} : \Upsilon_1,\ t : C \qquad y : D \;\Rightarrow\; \pi_2 : \bullet \mid \overline{\kappa} : \Upsilon_2}{E \;\vdash\; \pi_1, \pi_2\{y := \mathtt{y}(t)\} : \bullet \mid \overline{\ell} : \Upsilon_1, \overline{\kappa}\{y := \mathtt{y}(t)\} : \Upsilon_2\ , \mathtt{mkc}(t, \mathtt{y}) : C \smallsetminus D}$$
$\smallsetminus$ *left*:
$$\frac{x : C \;\vdash\; \pi_1 : \bullet \mid \overline{\ell} : \Upsilon, u : D}{y : C \smallsetminus D \;\vdash\; \pi_1 : \bullet, \mathtt{postp}(\mathtt{x} \mapsto u\{x := \mathtt{x}(y)\}, y) : \bullet \mid \overline{\ell}\{x := \mathtt{x}(y)\} : \Upsilon}$$
$\wp$ *right*
$$\frac{x : C \;\vdash\; \pi_1 : \bullet \mid \overline{\ell} : \Upsilon, t_0 : D_0, t_1 : D_1}{y : C \smallsetminus D \;\Rightarrow\; \pi_1 : \bullet, \overline{\ell} : \Upsilon, t_0 \wp t_1 : D_0 \wp D_1}$$
$\wp$ *left:*
$$\frac{x : D_0 \;\vdash\; \pi_1 : \bullet \mid \overline{\ell} : \Upsilon_1 \qquad y : D_1 \;\Rightarrow\; \pi_2 : \bullet \mid \overline{\kappa} : \Upsilon_2}{z : D_0 \wp D_1 \;\vdash\; \pi_1\{x := \mathtt{casel}(z)\}, \pi_2\{y := \mathtt{caser}(z)\} : \bullet \mid \overline{\ell}\{x := \mathtt{casel}(z)\} : \Upsilon_1, \overline{\kappa}\{y := \mathtt{caser}(z)\} : \Upsilon_2}$$

Table 2. Labelled sequent calculus $\mathbf{MLJ}^{\smallsetminus \wp}$

free variables x become bound in different parts of the computational context. Therefore to avoid "capture of free variables" in the term assignment to derivations in $\mathbf{NJ}^{\smallsetminus \curlyvee}$, it is enough to require derivatons to have the *pure parameter property*:

Convention. We assume that *in a derivation free variables assigned to distinct open assumptions are distinct.*

Remark 4. Consider the *global reduction* in equation 4: in a distributed interpretation, after the substitution $u\{z := t\}$ has been performed in the location of u, the substitutions $\overline{\zeta}\{y := u\{z := t\}\}$ and $\overline{\xi}\{z := t\}$ must be performed in remote locations, which are reachable because the term assignment encodes the structure of a *tree like* network (see in particular the graphical representation in the Appendix, section 8.3). Remote substitutions evoke the notion of *agent mobility* which is crucial to *distributed* versions of the π-calculus.

6.1 Labelled Prawitz' trees

As trees in Prawitz style Natural Deduction $\mathbf{MNJ}^{\multimap}$ can be decorated with linear λ terms, so we find more perspicuous to assign terms of our dual linear calculus to Prawitz trees for $\mathbf{MNJ}^{\smallsetminus}$ derivations, rather than to $\mathbf{MLJ}^{\smallsetminus}$ sequent derivations. In the linear case the correspondence between the sequent calculus $\mathbf{co} - \mathbf{MLJ}^{\smallsetminus}$ and natural deduction $\mathbf{MNJ}^{\smallsetminus}$ is straightforward. For the general case see [4].

In the Appendix (Section 8) we consider part of the derivation *with cut* in our example (Sections 3.2 and 8.2) and give the assignment of terms of the dual linear calculus to *Prawitz' trees* for $\mathbf{MNJ}^{\smallsetminus}$.

6.2 Dualities

Given the Natural Deduction system for $\mathbf{MNJ}^{\multimap\otimes}$ with term assignment in the linear lambda calculus with $\otimes$ and `let` operators, as in [7], consider the Natural Deduction system for $\mathbf{MNJ}^{\smallsetminus\wp}$ with term assignment corresponding to the sequent calculus in Tables 2. In this setting the following facts are clear:

Proposition 1. (*i*) *Given a Natural Deduction derivation* $\mathcal{D}_S$ *of a sequent with term assignment* $S : x_1 : A_1, \ldots, x_n : A_n \vdash t : A$ *in* $\mathbf{MNJ}^{\multimap\otimes}$*, there is a Natural Deduction derivation* $\mathcal{D}^{\perp}_{S^{\perp}}$ *of a sequent with term assignment* $S^{\perp} : x : A^{\perp} \vdash t_1 : A_1^{\perp}, \ldots, t_n : A_n$ $\mathbf{IMLJ}^{\smallsetminus\wp}$*, and conversely;*
(*ii*) *If a* $\mathbf{MNJ}^{\multimap\otimes}$ *derivation* $\mathcal{D}_S$ *reduces to* $\mathcal{D}'_{S'}$*, then the* $\mathbf{MNJ}^{\smallsetminus\wp}$ *derivation* $\mathcal{D}^{\perp}_{S^{\perp}}$ *reduces in one step to* $(\mathcal{D}')^{\perp}_{(S')^{\perp}}$*, and conversely.*

The proposition is proved by a straightforward induction on the length of the given derivation (part (i)) and on the length of the reduction sequence (part (ii)). It is understood that "a step" of reduction in the dual linear calculus must be seen as "macro" instruction for several steps of rewriting, which may nevertheless be seen as a unit.

In conclusion, reconsider the question *is the dual linear calculus a distributed system?* This is a non-mathematical question: the answer depends on opinions of what a distributed system is. However, evidence gathered so far shows that Gentzen calculi for co-intuitionistic linear logic and the dual linear calculus certainly admit a distributed interpretation, in a sense that bears some resemblance to the meaning of "distributed" as used in "distributed π-calculus".

7 Conclusion

In this paper we have revisited a version of Milner's *synchronous π-calculus* that was used in the early days of the subject as documented in [6]; as a test case we have applied Abramsky's translation of classical multiplicative linear logic into the synchronous π-calculus to *co-intuitionistic linear logic* **co-IMLL**$^{\smallsetminus\wp}$ with *subtraction* and *par*. For this logic we have presented Gentzen systems which are dual to the corresponding systems for intuitionistic multiplicative linear logic **IMLL**$^{\multimap\otimes}$ with *linear implication* and *tensor product* and also we have given a term assignment to those Gentzen systems for **co-IMLL**$^{\smallsetminus\wp}$ in a *dual linear calculus*. We chose linear co-intuitionistic logic because it is naturally formalized in *multiple-conclusion* sequents and our calculus assigns a term to each formula in the succedent of a sequent, thus suggesting an interpretation of the term assignment as a distributed system, where each formula inhabits a distinct location. Having compared the translation in the synchronous π-calculus and the dual linear term assignment applied to a derivation that encodes a simple Petri Net computation, we concluded that the dual linear term assignment expresses the intuition of a distributed computation more clearly than the synchronous π-calculus.

This suggests comparing the dual linear term assignment with a translation of linear co-intuitionistic logic in some system of *distributed* π-calculus. The general lines of such translation seem clear: distinct subprocesses should be assigned to the active formulas of a *subtraction introduction* and they should lie in distinct locations; similarly, a subprocess should be assigned to the active formulas of a *subtraction elimination* and it should lie in a distinct location from those of the subprocesses assigned to the passive formulas; similar considerations may apply to subtraction. Some substitutions involved in the cut-elimination process may be performed in a locations, but others will have to be routed to remote locations. However the choice of a suitable distributed π-calculus and the details of such a translation may require some thought.

Moreover, one would like to consider the full system of co-intuitionistic logic **co-IL**$^{\smallsetminus\wp}$, with the sequent calculus **LJ**$^{\smallsetminus\curlyvee}$, not just its linear part. Here we have the structural rules Weakening and Contraction, and the π-calculus translation uses the iteration operator; as a consequence, the simple step by step correspondence between cut-elimination and π-calculus reduction no longer holds and one need to reason modulo an appropriate notion of bisimulation. It seems appropriate to leave such a project to future work.

References

1. R. M. Amadio. An asynchronous model of locality, failure, and process mobility, In *COORDINATION '97, Lecture Notes in Computer Science* volume 1282, Springer Verlag, 1997, pp. 374–391.
2. G. Bellin. Chu's Construction: A Proof-theoretic Approach. in Ruy J.G.B. de Queiroz editor, *Logic for Concurrency and Synchronisation*, Kluwer Trends in Logic n.18, 2003, pp. 93-114.
3. G. Bellin. A Term Assignment for Dual Intuitionistic Logic, conference paper presented at the LICS'05-IMLA'05 Workshop, Chicago, IL, June 30, 2005

4. G. Bellin. Assertions, hypotheses, conjectures and expectations. Rough set semantics and proof-theory, in: *Advances in Natural Deduction* - Proceedings of the Natural Deduction conference in Rio de Janeiro, 2001, Kluwer Trends in Logic, to appear 2011.
5. G. Bellin, M. Hyland, E.Robinson and C. Urban. Categorical Proof Theory of Classical Propositional Calculus *Theoretical Computer Science* Vol. 364, 2, November 2006, pp. 146-165.
6. G. Bellin and P. J. Scott. On the Pi-calculus and linear logic, *Theoretical Computer Science* 135, 1994, pp. 11-65.
7. G. Bierman *On Intuitionistic Linear Logic*, PhD Dissertation, University of Cambridge Computer Laboratory, Technical Report n. 346,1994
8. T. Crolard. A Formulae-as-Types Interpretation of Subtractive Logic. In: *Journal of Logic and Computation*, vol.14(4), 2004, pp. 529-570
9. J-Y. Girard. Linear Logic, *Theoretical Computer Science* 50, 1987, pp. 1-102.
10. M. Hennessy. *A distributed pi-calculus*, Cambridge University Press, 2007.
11. M. Hennessy, M. Merro and J. Rathke, Towards a behavioural theory of access and mobility control in distributed systems, *Theoretical Computer Science* 322, (3), 2004, pp. 615-669
12. M. Hennessy and J. Riely. Resource Access Control in Systems of Mobile Agents. *Information and Computation*, 173 (1). 2002, pp. 82-120
13. J. M. Hyland and V. de Paiva. Full Intuitionistic Linear Logic, *Annals of Pure and Applied Logic* 64, 1993, pp. 273-91.
14. F. Lamarche. Proof Nets for Intuitionistic Linear Logic: Essential Nets, 2008 Available from `http://hal.inria.fr/inria-00347336/`
15. C. Laneve and B. Victor. Solos in Concert, in: *Mathematical Structures in Computer Science* 13, 5, 2003, pp. 657-683.
16. N. Martí-Oliet and J. Meseguer. From Petri Nets to Linear Logic, in *Category Theory and Computer Science, Manchester, UK.* Springer-Verlag *LNCS* 389, 1989.
17. M. Merro. On the observational theory of the CPS-calculus. *Acta Informatica* 47, 2010, pp. 111–132.
18. R. Milner. The polyadic π-calculus: A tutorial, in: Bauer, Brauer and Schwichtenberg eds., *Logic and Algebra of Specification*, vol. 94 of *Series F.* NATO ASI, Springer 1993.
19. P. Sewell. Global/local subtyping and capability inference for a distributed π-calculus, In *Proceedings of ICALP '98, LNCS 1443*, 1998, Springer-Verlag, pp. 695–706.
20. H. Thielecke. *Categorical Structure of Continuation Passing Style*. PhD Thesis, University of Edinburgh, 1997. Also available as technical report ECS-LFCS-97-376.

8 APPENDIX. Examples of computation.

In Section 8.1 of this Appendix we give the derivation $\mathcal{D}_0^\perp$ *with cuts* in multiplicative co-intuitionistic $\mathbf{MLJ}^{\setminus\curlyvee}$, which is dual of the derivation $\mathcal{D}_0$ in multiplicative linear logic in Section 3.1: both $\mathcal{D}_0$ and $\mathcal{D}_0^\perp$ represent a strategy for the Petri net computation in Figure 1.

Next we show two encodings of a part of the derivation given in Section 8.1 and of the cut-elimination process applied to it: in Section 8.2 we use the synchronous π-calculus and in Section 8.3 we use our dual calculus.

8.1 Example: derivation with cut in the sequent calculus $\mathbf{MLJ}^{\setminus\curlyvee}$

Here are the dual derivations $\mathcal{D}_{11}^{\perp}$

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{E \vdash E \qquad D \vdash D}{E \vdash E \setminus D, D}\setminus\text{-R} \qquad \dfrac{\dfrac{C \vdash C \qquad B \vdash B}{C \vdash C \setminus B, B}\setminus\text{-R} \qquad A \vdash A}{C \vdash (C \setminus B) \setminus A, A, B}\setminus\text{-R}}{E \vdash (E \setminus D) \setminus C, T_1^{\perp}, A, B, D}\setminus\text{-R}}{E \setminus B \vdash T_1^{\perp}, T_2^{\perp}, A, D}}{(E \setminus B) \setminus D \vdash T_1^{\perp}, T_2^{\perp}, A}}{((E \setminus B) \setminus D) \setminus A \vdash T_1^{\perp}, T_2^{\perp}}
$$

$\mathcal{D}_{12}^{\perp}$:

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{E \vdash E \qquad B \vdash B}{E \vdash B, E \setminus B} \qquad D \vdash D}{E \vdash B, D, (E \setminus B) \setminus D} \qquad A \vdash A}{E \vdash A, B, D, ((E \setminus B) \setminus D) \setminus A} \qquad \dfrac{A \vdash A \qquad D \vdash D}{A \wp D \vdash A, D}}{E \wp (A \wp D) \vdash A, A, B, D, D, ((E \setminus B) \setminus D) \setminus A}}{E \wp (A \wp D) \vdash A \wp A, B, D, D, ((E \setminus B) \setminus D) \setminus A}}{E \wp (A \wp D) \vdash A \wp A, B, D \wp D, ((E \setminus B) \setminus D) \setminus A}
$$

$\mathcal{D}_{21}^{\perp}$ and $\mathcal{D}_{22}^{\perp}$:

$$
\dfrac{A \vdash A \qquad A \vdash A}{A \wp A \vdash A, A} \qquad \dfrac{D \vdash D \qquad D \vdash D}{D \wp D \vdash D, D}
$$

Now the derivation $\mathcal{D}_0^{\perp}$ has cut formulas $\mathbf{C}_1 = ((E \setminus B) \setminus D) \setminus A$, $\mathbf{C}_2 = D \wp D$ and $\mathbf{C}_3 = A \wp A$:

$$
\dfrac{\dfrac{\dfrac{\begin{matrix}\mathcal{D}_{12}^{\perp} \\ E \wp (A \wp D) \vdash A \wp A, D \wp D, \mathbf{C}_1\end{matrix} \qquad \begin{matrix}\mathcal{D}_{11}^{\perp} \\ \mathbf{C}_1 \vdash T_1^{\perp}, T_2^{\perp}\end{matrix}}{E \wp (A \wp D) \vdash T_1^{\perp}, T_2^{\perp} A \wp A, B, D \wp D}\mathit{cut}_1 \qquad \begin{matrix}\mathcal{D}_{22}^{\perp} \\ D \wp D \vdash D, D\end{matrix}}{E \wp (A \wp D) \vdash T_1^{\perp}, T_2^{\perp} A \wp A, B, D, D}\mathit{cut}_2 \qquad \begin{matrix}\mathcal{D}_{21}^{\perp} \\ A \wp A \vdash A, A\end{matrix}}{E \wp (A \wp D) \vdash T_1^{\perp}, T_2^{\perp} A, A, B, D, D}\mathit{cut}_3
$$

8.2 Example in the π-calculus

We go through the π-calculus translation of the proofs $\mathcal{D}_{11}^{\perp}$ and $\mathcal{D}_{12}^{\perp}$ in Section 8.1 step by step.

Derivation $\mathcal{D}_{11}^{\perp}$.

First we have four steps of *subtraction introduction* ($\searrow$-R).

$$\pi(\mathbf{st}_1) = \left\{ \frac{I_{ee_1}:\ e:E \vdash e_1:E \qquad I_{dd_1}:\ d:D \vdash d_1:D}{\mathbf{SI}_{z_1}^{e_1 d}(I_{ee_1} \| I_{dd_1}):\ e:E \vdash z_1: E \searrow D, d_1:D} \right.$$

$$\pi(\mathbf{st}_{2,3}) = \left\{ \frac{\dfrac{I_{cc_1}:\ c:C \vdash c_1:C \qquad I_{bb_1}:\ b:B \vdash b_1:B}{\mathbf{SI}_{z_2}^{c_1 b}(I_{cc_1} \| I_{bb_1}):\ c:C \vdash z_2: C \searrow B, b_1:B} \qquad I_{aa_1}:\ a:A \vdash a_1:A}{\mathbf{SI}_{z_3}^{z_2 a}\Big(\mathbf{SI}_{z_2}^{c_1 b}(I_{cc_1} \| I_{bb_1}) \| I_{aa_1}\Big):\ c:C \vdash z_3: \underbrace{(C \searrow B) \searrow A}_{T_1^{\perp}}, a_1:A, b_1:B} \right.$$

$$\pi(\mathbf{st}_4) = \left\{ \frac{\pi(\mathbf{st}_1):\ e:E \vdash z_1:E \searrow D, d_1:D \qquad \pi(\mathbf{st}_{2,3}):\ c:C \vdash z_3:T_2, a_1:A, b_1:B}{\mathbf{SI}_{z_4}^{z_1 c}\Big(\pi(\mathbf{st}_1) \| \pi(\mathbf{st}_{2,3})\Big):\ e:E \vdash z_4: \underbrace{(E \searrow D) \searrow C}_{T_2^{\perp}}, z_3:T_1^{\perp}, a_1:A, b_1:B, d_1:D} \right.$$

Next we have three steps for *subtraction elimination* ($\searrow$-L)

$$\pi(\mathcal{D}_{11}^{\perp}) = \left\{ \frac{\pi(\mathbf{st}_4):\ e:E \vdash z_3:T_1^{\perp}, z_4:T_2^{\perp}, a_1:A, b_1:B, d_1:D}{\mathbf{SE}_{z_7}^{z_6 a_1}\Big(\mathbf{SE}_{z_6}^{z_5 d_1}\big(\mathbf{SE}_{z_5}^{e b_1}\pi(\mathbf{st}_4)\big)\Big):\ z_7:((E \searrow B) \searrow D) \searrow A \vdash z_3:T_1^{\perp}, z_4:T_2^{\perp}} \right.$$

Derivation $\mathcal{D}_{12}^{\perp}$:

Again we have three steps of *subtraction introduction* ($\searrow$-R).

$$\pi(\mathbf{intro}) = \left\{ \begin{array}{l} \mathbf{SI}_{z_{10}}^{z_9 a''}\ \Big(\mathbf{SI}_{z_9}^{z_8 d''}\ \big(\mathbf{SI}_{z_8}^{e"_1 b''}\ (I_{e''e''_1} \| I_{b''b''_1}) \big\| I_{d''d''_1}\big) \Big\| I_{a''a''_1}\Big): \\ \qquad e'':E \vdash a''_1:A, b''_1:B, d''_1:D, z_{10}:((E \searrow B) \searrow D) \searrow A \end{array} \right.$$

Next we have two steps of *par elimination* ($\wp$-L)

$$\pi(\mathbf{parelim}_1) = \mathbf{DE}_{w_1}^{a^+ d^+}\big(I_{a^+ a_1^+} \| I_{d^+ d_1^+}\big) : w_1 : A \wp D \vdash a_1^+ : A, d_1^+ : D$$

and

$$\pi(\mathbf{parelim}_2) = \left\{ \begin{array}{c} \pi(\mathbf{intro})_{e" a"_1 b"_1 d"_1 z_1 0} \qquad \pi(\mathbf{parelim}_1)_{w_1 a_1^+ d_1^+} \\ \hline \mathbf{DE}_{w_2}^{e" w_1}\big(\pi(\mathbf{intro}) \| \pi(\mathbf{parelim}_1)\big): \\ w_2 : E \wp (A \wp D) \vdash a"_1, a_1^+ : A, b"_1 : B, d"_1, d_1^+ : D, \\ z_{10} : ((E \searrow B) \searrow D) \searrow A \end{array} \right.$$

followed by two steps of *par introduction* ($\wp$-R)

$$\pi(\mathcal{D}_{12}) = \begin{cases} \dfrac{\pi(\mathbf{parelim}_2)_{w_2 a"_1 a_1^+ d"_1 d_1^+ z_{10}}}{\mathbf{DE}_{w_4}^{a"_1 a^+1}\Big(\mathbf{DE}_{w_3}^{d"_1 d_1^+}\left(\pi(\mathbf{parelim}_2)\right)\Big)} \\ w_2 : E\wp(A\wp D) \vdash w_4 : A\wp A, b"_1 : B, w_3 : D\wp D, z_{10} : ((E \setminus B) \setminus D) \setminus A \end{cases}$$

Consider the π-calculus translation of the result of cut_1 with cut formula $\mathbf{C}_1 = ((E \setminus B) \setminus D) \setminus A$, namely,

$$P = (\nu z)\mathcal{D}_{12}^{\perp}[z/z_{10}] \parallel \mathcal{D}_{11}^{\perp}[z/z_7]$$

The processes $\pi(\mathcal{D}_{12}^{\perp})[z/z_{10}]$ and $\pi(\mathcal{D}_{11}^{\perp})[z/z_7]$ can communicate only through the channel z; their interaction triggers interactions between z_9 and z_6 (renamed z_9), and between between z_8 and z_5 (renamed z_8); in these steps also e, a_1, b_1 and d_1 in $\pi(\mathcal{D}_{11}^{\perp})$ have been renamed as e''_1, $a"$, $b"$ and $d"$. This makes it possible to have interactions between the subprocesses $I_{e''_1 e_1}$, $I_{aa"}$, $I_{dd"}$, $I_{bb"}$ (after renaming) of $\pi(\mathcal{D}_{11}^{\perp})$ with the subprocesses $I_{e''e''_1}$, $I_{a"a"_1}$, $I_{d"d"_1}$ $I_{b"b"_1}$ of $\pi(\mathcal{D}_{11}^{\perp})$, respectively: the resulting rewritings yield subprocesses $I_{e''e_1}$, $I_{aa"_1}$, $I_{dd"_1}$ and $I_{bb"_1}$ and corresponds to the cut-elimination process applied to cut_1. Performing the formal steps briefly described here we see that the process P reduces to

$$P^* = \left(\mathbf{SI}_{z_4}^{z_1 c}\Big(\mathbf{SI}_{z_1}^{e_1 d}(I_{e''e_1} \| I_{dd''_1})\Big\|\mathbf{SI}_{z_3}^{z_2 a}\Big(\mathbf{SI}_{z_2}^{c_1 b}(I_{cc_1} \| I_{bb''_1}) \;\Big\|\; I_{aa''_1}\Big)\Big)\right) : \\ e'' : E \vdash z_4 : T_2^{\perp}, z_3 : T_1^{\perp}, a''_1 : A, b''_1 : B, d''_1 : D$$

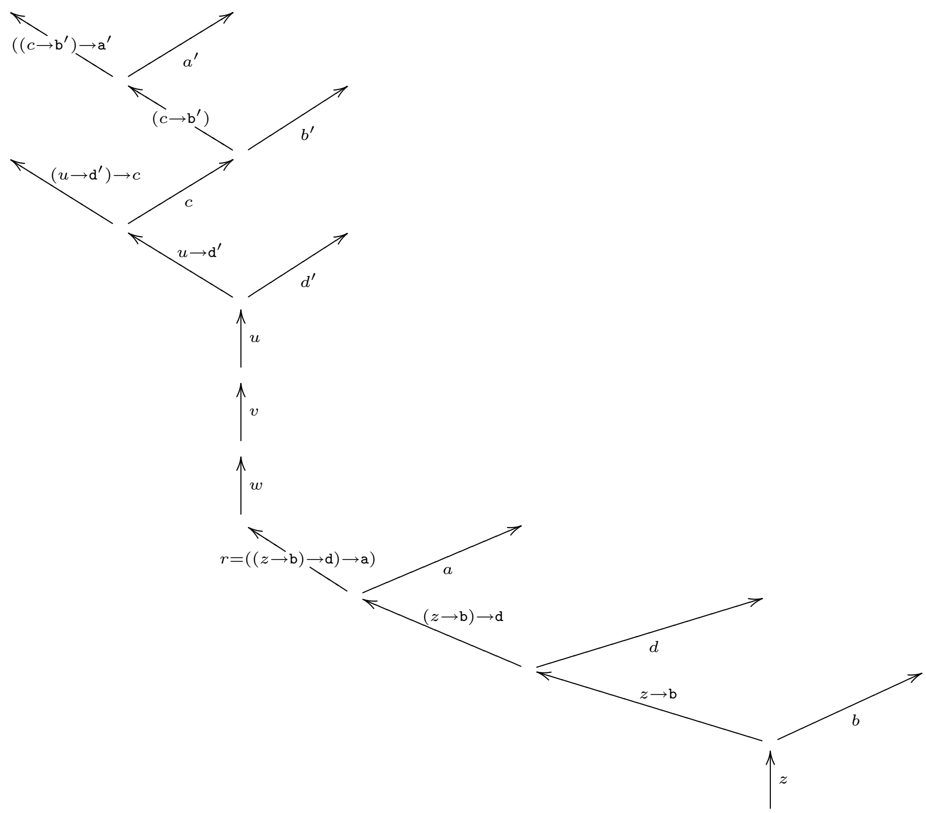
The rest of the proof is similar and is omitted.

8.3 Example in the dual linear calculus

In this section we give the term assignment in the *dual linear calculus* to a part of the $\mathbf{MLJ}^{\setminus \curlyvee}$ derivation given in Section 8.1.

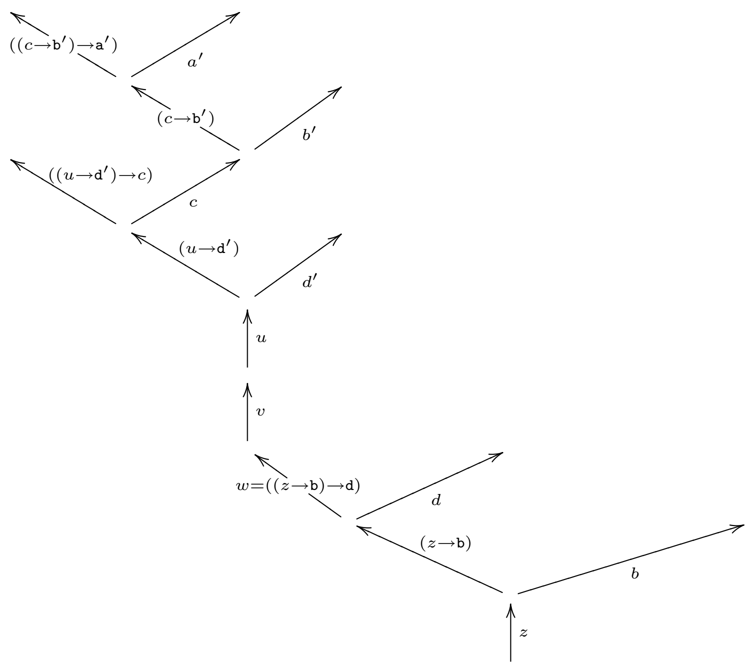
For convenience, we still draw trees with the root at the bottom, keeping in mind that here derivations are built from bottom up. We shall use the notation $(t \to \mathtt{a})$ for $\mathtt{mkc}(t, a)$ and $\xrightarrow{e \mapsto u} t$ for $\mathbf{postp}(e \mapsto u, t)$.

$$\mathcal{S}_0 : \xleftarrow[\mathcal{R}ed_0]{\mathtt{w}\mapsto a'} r \quad \xleftarrow{\mathtt{u}\mapsto b'} v, \quad \xleftarrow{\mathtt{v}\mapsto d'} w, \qquad ((c \to b') \to a') \qquad ((u \to d') \to c)$$



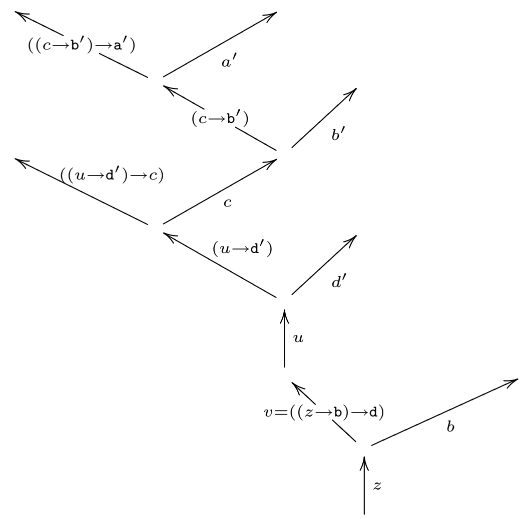
reduces to

$\mathcal{S}_1 : \xleftarrow[\mathcal{R}ed_1]{\mathtt{v}\mapsto d'} w \quad \xleftarrow{\mathtt{u}\mapsto b'} v,$ $((c \to b') \to a')$ $((u \to d') \to c)$



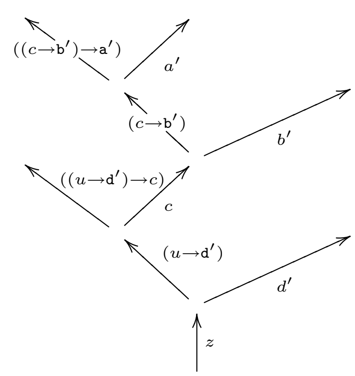
reduces to

$\mathcal{S}_2 : \xleftarrow[\mathcal{R}ed_2]{\mathtt{u}\mapsto b'} (z \to \mathtt{b}$ $\quad ((c \to b') \to a') \quad ((u \to d') \to c)$



reduces to

$\mathcal{S}_3 :$ $\quad ((c \to b') \to a') \quad ((u \to d') \to c)$



We show here the steps of the computation:

$\mathcal{S}_0 :$ $\xleftarrow[\mathcal{R}ed_0]{\mathtt{w}\mapsto a'} r,$ $\xleftarrow{\mathtt{u}\mapsto b'} v,$ $\xleftarrow{\mathtt{v}\mapsto d'} w,$ $((c \to b') \to a')$ $((u \to d') \to c)$

where $b = \mathbf{b}(z),$ $d = \mathbf{d}(z \to \mathtt{b}),$ $a = \mathbf{a}((z \to \mathtt{b}) \to \mathtt{d}),$ $w = \mathbf{w}((((z \to \mathtt{b}) \to \mathtt{d}) \to \mathtt{a})),$ $v = \mathbf{v}(w)$
$u = \mathtt{u}(v),$ $d' = \mathbf{d}'(u),$ $c = \mathbf{c}(u \to \mathtt{d}'),$ $b' = \mathbf{b}'(c),$ $a' = \mathbf{a}(c \to \mathtt{b}').$

Reducing $\mathcal{R}ed_0$**:** $\mathcal{S}_1 = \mathcal{S}_0 - \mathcal{R}ed_0 \quad \{w := ((z \to \mathtt{b}) \to \mathtt{d})\}\ \{a := a'\{w := ((z \to \mathtt{b}) \to \mathtt{d})\}\}$

$\mathcal{S}_1 :$ $\xleftarrow[\mathcal{R}ed_1]{\mathtt{v}\mapsto d'} ((z \to \mathtt{b}) \to \mathtt{d}),$ $\xleftarrow{\mathtt{u}\mapsto b'} v,$ $((c \to b') \to a')$ $((u \to d) \to c)$

where $b = \mathbf{b}(z),$ $d = \mathbf{d}(z \to \mathtt{b}),$ $v = \mathbf{v}((z \to \mathtt{b}) \to \mathtt{d})$
$u = \mathtt{u}(v),$ $d' = \mathbf{d}'(u),$ $c = \mathbf{c}(u \to \mathtt{d}'),$ $b' = \mathbf{b}'(c),$ $a' = \mathbf{a}(c \to \mathtt{b}').$

Reducing $\mathcal{R}ed_1$**:** $\mathcal{S}_2 = \mathcal{S}_1 - \mathcal{R}ed_1 \quad \{v := (z \to \mathtt{b})\}\ \{d := d'\{v := (z \to \mathtt{b})\}\}.$

$\mathcal{S}_2 :$ $\xleftarrow{\mathtt{y}\mapsto (b)} e,$ $\xleftarrow[\mathcal{R}ed_2]{\mathtt{u}\mapsto b'} (z \to \mathtt{b})$ $((c \to b') \to a')$ $((u \to d) \to c)$ (ii)

where $b = \mathbf{b}(z),$ $u = \mathtt{u}((z \to \mathtt{b})),$ $d' = \mathbf{d}'(u),$ $c = \mathbf{c}(u \to \mathtt{d}'),$ $b' = \mathbf{b}'(c),$ $a' = \mathbf{a}(c \to \mathtt{b}').$

Reducing $\mathcal{R}ed_2$**:** $\mathcal{S}_3 = \mathcal{S}_2 - \mathcal{R}ed_2 \quad \{u := z\}\ \{b := b'\{u := z\}\}.$

$\mathcal{S}_3 :$ $((c \to b') \to a')$ $((u \to d) \to c)$ (iii)

where $d' = \mathbf{d}'(z),$ $c = \mathbf{c}(z \to \mathtt{d}'),$ $b' = \mathbf{b}'(c),$ $a' = \mathbf{a}(c \to \mathtt{b}').$