Polarised Intermediate Representation of Lambda Calculus with Sums

Guillaume Munch-Maccagnoni, Gabriel Scherer

Polarised Intermediate Representation of Lambda Calculus with Sums

Guillaume Munch-Maccagnoni
Computer Laboratory, University of Cambridge

Gabriel Scherer
Gallium, Inria Paris-Rocquencourt

***Abstract*—The theory of the λ-calculus with extensional sums is more complex than with only pairs and functions. We propose an untyped representation—an *intermediate calculus*—for the λ-calculus with sums, based on the following principles: 1) Computation is described as the reduction of pairs of an expression and a context; the context must be represented inside-out, 2) Operations are represented abstractly by their transition rule, 3) Positive and negative expressions are respectively eager and lazy; this polarity is an approximation of the type. We offer an introduction from the ground up to our approach, and we review the benefits.**

A structure of alternating phases naturally emerges through the study of normal forms, offering a reconstruction of focusing. Considering further purity assumption, we obtain *maximal multi-focusing*. As an application, we can deduce a syntax-directed algorithm to decide the equivalence of normal forms in the simply-typed λ-calculus with sums, and justify it with our intermediate calculus.

I. Introduction

The simply-typed λ-calculus with *extensional* (or *strong*) *sums*, in equational form, is recalled in Figure 1.

a) The rewriting theory of this calculus is more complex than with pairs and functions alone: As is well-known, the reduction relation on open terms requires additional commutations, recalled in Figure 2. But the complexity is also witnessed by the diversity of approaches to decide the $\beta\eta$-equivalence of terms, proposed since 1995: based either on a non-immediate rewriting theory (Ghani [23]; Lindley [45]) or on normalisation by evaluation (Altenkirch, Dybjer, Hofmann, and Scott [3]; Balat, di Cosmo, and Fiore [6]).

Moreover, syntactic approaches are delicate, because there can be no λ-calculus with sums *à la* Curry. Indeed, recall that cartesian closed categories with binary co-products and arbitrary fixed points are inconsistent (Lawvere [40]; Huwig and Poigné [33]). For the calculus in Figure 1, this means that, in the absence of typing constraints, any two terms are equi-convertible (by a diagonal argument involving the fixed point of the function $\lambda x.\delta(x, y.\iota_2(y), y.\iota_1(y))$, see Dougherty [16] for details).

b) On the side of proof theory: To begin with, sums become as simple as products once cast into Gentzen's sequent calculus [22], perhaps because sequent calculus has an inherent symmetry that reflects the categorical duality between sums and products. For instance, commutations are not needed for reducing proofs. Thanks to *abstract-machine-like* calculi in the style of Curien and Herbelin [11], we can transfer the benefits of sequent calculus to term calculi.

The *focusing* discipline of proof theory provides a characterisation of cut-free, η-long proofs (Andreoli [4]). It has been applied to intuitionistic logic with sums by Liang and Miller [43, 44]. We continue the first author's approach to focalisation [47], where the properties related to focusing are expressed as constraints to the reduction, rather than as restrictions to the rules of the system themselves.

$$t, u, v ::= x \mid \lambda x.t \mid t\,u \mid \langle t; u\rangle \mid \pi_1(t) \mid \pi_2(t) \mid \iota_1(t) \mid \iota_2(t) \mid \delta(t, x.u, y.v)$$
$$A, B ::= X \mid A \to B \mid A \times B \mid A + B$$

(a) Terms and types

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma \vdash t : A_i}{\Gamma \vdash \iota_i(t) : A_1 + A_2}\ i \in \{1,2\}$$
$$\frac{\Gamma \vdash t : A_1 + A_2 \qquad \Gamma, x_i : A_i \vdash u_i : B\ (\forall i \in \{1,2\})}{\Gamma \vdash \delta(t, x_1.u_1, x_2.u_2) : B}$$
$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash u : B}{\Gamma \vdash \langle t; u\rangle : A \times B} \qquad \frac{\Gamma \vdash t : A_1 \times A_2}{\Gamma \vdash \pi_i(t) : A_i}\ i \in \{1,2\}$$
$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B} \qquad \frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\,u : B}$$

(b) Typing rules

$$(\lambda x.t)\,u \approx t[u/x] \qquad \pi_i(\langle t_1; t_2\rangle) \approx t_i$$
$$\delta(\iota_i(t), x_1.u_1, x_2.u_2) \approx u_i[t/x_i]$$

(c) β-reductions

$$t \approx \lambda x.(t\,x) \qquad t \approx \langle \pi_1(t); \pi_2(t)\rangle$$
$$u[t/y] \approx \delta(t, x_1.u[\iota_1(x_1)/y], x_2.u[\iota_2(x_2)/y])$$

(d) η-expansions

Figure 1 – λ-calculus with sums

Our approach is inspired by Girard's *polarisation* [24], which was introduced to circumvent another inconsistency of cartesian closed categories, in the presence of natural isomorphisms $A \cong (A \to R) \to R$. It consists in formally distinguishing the positive connectives from the negative ones, and in letting the reduction order be locally determined by this polarity, as it was later understood by Danos, Joinet, and Schellinx [13], and others (see [49]). In categorical terms, polarisation relaxes the hypothesis that composition is associative, in a meaningful way [50].

c) Contents of the article: In Section III, we combine abstract-machine-like calculi, polarisation, and the direct approach to focusing, to propose a Curry-style calculus with extensional sums. In this article we refer to this new calculus as $\mathbf{L}_{\text{int}}$. We show that the λ-calculus with sums is isomorphic to *depolarised* $\mathbf{L}_{\text{int}}$.

Then, in Section IV we describe a syntax-directed algorithm for deciding when untyped normal forms in $\mathbf{L}_{\text{int}}$ are equivalent modulo normalization-preserving conversions. Then, assuming depolarisation, we refine the algorithm to decide the equivalence of normal forms in the simply-typed λ-calculus with sums in a syntax-directed manner.

Before this, in Section II, we introduce and motivate the

$$(\lambda x.t)\,u \succ t[u/x] \qquad \pi_i(<t_1;t_2>) \succ t_i$$
$$\delta(\iota_i(t), x_1.u_1, x_2.u_2) \succ u_i[t/x_i]$$

(a) Main reductions

$$\delta(t, x_1.u_1, x_2.u_2)\,v \succ \delta(t, x_1.(u_1\,v), x_2.(u_2\,v))$$
$$\pi_i(\delta(t, x_1.u_1, x_2.u_2)) \succ \delta(t, x_1.\pi_i(u_1), x_2.\pi_i(u_2))$$
$$\delta_v(\delta(t, x_1.u_1, x_2.u_2)) \succ \delta(t, x_1.\delta_v(u_1), x_2.\delta_v(u_2))$$
where $\delta_v(t) = \delta(t, y_1.v_1, y_2.v_2)$.

(b) Commutations

Figure 2 – Reduction relation for the λ-calculus with sums

technique of abstract-machine-like calculi as it now stands. The novelty of this exposition is to show that these calculi derive from simple principles. Thanks to these principles, the article is rather self-contained, and furthermore in Section V we can better underline the differences with the related techniques of focusing and continuation-passing style (CPS).

A. Abstract-machine-like calculi (Section II)

Term syntax can benefit from the symmetry of sequent calculus by following certain principles, such as *treating seriously the duality between expressions and contexts*, as discovered with Curien and Herbelin's abstract-machine-like calculi [11]. These calculi describe computation as the reduction of an expression against a context, the latter being described in a grammar of its own.

Abstract-machine-like calculi evidence a correspondence between sequent calculus (Gentzen [22]), abstract machines (Landin [37, 38]) and CPS (Van Wijngaarden [65] and others). In particular they illustrate the links that CPS has with categorical duality (Filinski [21]; Thielecke [64]; Selinger [58]) and with abstract machines (Reus and Streicher [63]; Ager, Biernacki, Danvy, and Midtgaard [2]; Danvy and Millikin [14]). Curien-Herbelin's calculi are however set in classical logic and include variants of Scheme's *call/cc* control operator.

There are now abstract-machine-like calculi, with various applications, by various authors, that are based on the same technique [11, 66, 32, 5, 47, 12, 48, 50, 7, 17, 49, 60]. The first goal of the article is to adapt abstract-machine-like calculi to the λ-calculus with sums. We remove the full power of *call/cc* following an idea of Herbelin [32], already applied by Espírito Santo, Matthes, and Pinto [19] for the call-by-name λ-calculus. In doing this adaptation, we first advocate in Section II that the abstract-machine-like calculi can be summarised with two principles: *the inside-out representation of contexts is primitive* and *language constructs are the abstract solutions of equations given by their machine transitions*. These principles lead us to a first calculus with sums in call-by-name referred to in this article as $\mathbf{L}^{\ominus}_{\mathrm{int}}$.

Following the latter idea, not all the constructs of the λ-calculus with sums are given as primitives of $\mathbf{L}_{\mathrm{int}}$. But, they are all retrieved in a systematic way, as we will see. Therefore we advocate that abstract-machine-like calculi should be seen as *intermediate calculi* that reveal the hidden structure of the terms—an analogy with *intermediate representations* used by compilers, which enable program transformation and analysis. In fact, abstract-machine-like calculi can be described as *defunctionalised CPS in direct style*, as we will explain (Section V-C).

This exposition complements Wadler's introduction to term assignments for Gentzen's classical sequent calculus LK [66], Curien and the first author's reconstruction of LK from abstract machines [12], and Spiwak's motivation of abstract-machine-like calculi for the programming language theory [60].

We leave expansions aside until Section III.

B. Intuitionistic polarisation (Section III)

For our current purposes, polarisation corresponds to the principle that *an expression is either positive or negative depending on the type; this polarity determines whether it reduces strictly or lazily*.

We can assume that such polarities are involved in the validity of η-expansions in the λ-calculus with sums. Indeed, according to Danos, Joinet and Schellinx [13] (although in the context of classical sequent calculus), connectives can be distinguished upon whether the η-expansion seems to force or to delay the reduction. For instance, expanding $u[t/y]$ into $\delta(t, x_1.u[\iota_1(x_1)/y], x_2.u[\iota_2(x_2)/y])$ forces the evaluation of t. They show (with LK^{η}_{p}) how by making the reduction match, for each connective, the behaviour thus dictated by possible η-expansions, one essentially finds back polarisation in the sense of Girard.

This is why declaring sums positive (strict) and functions negative (lazy) ensures that η-expansions do not modify the evaluation order. Other criteria can determine the polarity of more complex connectives, but this is out of the scope of the current article (for instance, subject reduction in the case of quantifiers and modalities).

Polarities only matter when call by value and call by name differ. In the context of the pure λ-calculus, only non-termination can discriminate call by value from call by name. Polarisation therefore suggests a novel approach to typed λ-calculi where *associativity of composition is not seen as an axiom but as an external property, similar in status to normalisation*.

The polarised calculus $\mathbf{L}_{\mathrm{int}}$ is introduced in Section III, and it is a polarised abstract-machine-like calculus consistent with Girard's [25, 27] and Liang and Miller's [43, 44] suggestions for a focalised intuitionistic sequent calculus. (We go back on Liang and Miller's LJF in Section V-A.) $\mathbf{L}_{\mathrm{int}}$ also determines an intuitionistic variant of Danos, Joinet and Schellinx's LK^{η}_{p} [13]. It is also meant to be a direct-style counterpart to (a variant of) Levy's *Call-by-push-value* models [42], though a precise correspondence will be the subject of another work.

In the article, since we follow Curry's style, terms are not annotated by their types. The polarity is therefore also the least amount of information that has to be added to the calculus so that the evaluation order is uniquely determined, before any reference is made to the types. (We follow the same technique as appeared previously by the first author [47, 50, 49].)

Finally, we establish the relationship with the λ-calculus with sums under *depolarisation*, that is to say associativity of composition. This is the same notion of depolarisation as Melliès and Tabareau's [46] for linear logic, see [50].

C. Focusing, and deciding equivalence on normal forms (Section IV)

In this article, focalisation is obtained as an emergent property of the reduction of terms, rather than as a restriction to structure of the proofs. In Section IV, we show how normal forms then have a syntactic structure of alternating

phases of constructors and abstractors. This indeed corresponds to focused proof disciplines, with $\mathbf{L}_{\text{int}}$ systematically using abstractors to represent invertible rules, and constructors for non-invertible rules. (This usage recalls Zeilberger's analogy between invertibility and pattern-matching [69, 70], see Section V-B.)

We describe an algorithm that decides (normalisation-preserving) η-equivalence which is not type-directed, but which inspects the syntactic structure of normal forms. It is inspired by Abel and Coquand's techniques [10, 1], which deals with functions and (dependant) pairs, but not positive connectives like sums.

Then, variable and co-variable scoping fully determines the *independence* of neighbouring phases: if reordering two fragments of a normal form does not break any variable binding, it should be semantically correct. This is immediate for abstractor phases (which corresponds to the easy permutations of invertible steps), but requires explicit depolarisation assumptions for constructor phases. Rewriting phases according to their dependencies corresponds to the idea of *maximal multi-focusing* (Chaudhuri *et al.* [9, 8]; the second author discusses its relation with the $\beta\eta$-equivalence of sums in [57]). We give a fairly uniform syntactic definition of canonical forms as normal forms of phase-permuting and phase-merging rewriting relations. In particular, the $\beta\eta$-equivalence of typed λ-terms with sums can be decided by comparing these normal forms.

D. Notations

In the grammars that we define, a dot (.) indicates that variables before it are bound in what comes after. (For instance with $\mu(x{\cdot}\star).c$, the variables x and $\star$ are bound in c.)

If $\rhd$ is a rewriting relation, then the compatible closure of $\rhd$ is denoted by $\to$ and the compatible equivalence relation $(\leftarrow \cup \to)^*$ is denoted with $\simeq$. Reductions are denoted with $\rhd_{\mathrm{R}}$ and expansions with $\rhd_{\mathrm{E}}$. In this context we define $\rhd_{\mathrm{RE}} \stackrel{\text{def}}{=} \rhd_{\mathrm{R}} \cup \rhd_{\mathrm{E}}$, etc.

II. A Principled Introduction to Abstract-Machine-Like Calculi

A. Abstract machines

Abstract machines are defined by a grammar of *expressions* t, a grammar of *contexts* e, and rewriting rules on pairs $c = \langle t \parallel e \rangle$ (*commands*). A command $\langle\, t \parallel e \,\rangle$ represents the computation of t in the context e. Intuitively, contexts correspond to *expressions with a hole*: a distinguished variable $\square$ that appears exactly once. For instance, the following extension of the Krivine abstract machine decomposes a term such as $\delta((\lambda x.\iota_1(t))u, y.v, z.w)$ into the context $\delta(\square, y.v, z.w)$ and the redex $(\lambda x.\iota_1(t))u$. The abstract machine therefore determines an evaluation strategy—here, call by name—for reductions in Figure 1c.

Contexts are for now *stacks* S of the following form:

$$(e =)\ S ::= \star \mid u{\cdot}S \mid \pi_1{\cdot}S \mid \pi_2{\cdot}S \mid [x.u|y.v]{\cdot}S$$

The symbol $\star$ represents the empty context. The reduction relation $\rhd_{\mathrm{R}}$ on commands is defined by two sets of reduction rules:

- *Main reductions:* Variables are substituted, pairs are projected and branches are selected:

$$\langle \lambda x.t \parallel u{\cdot}S \rangle \rhd_{\mathrm{R}} \langle t[u/x] \parallel S \rangle$$
$$\langle <t_1 ; t_2> \parallel \pi_i{\cdot}S \rangle \rhd_{\mathrm{R}} \langle t_i \parallel S \rangle$$
$$\langle \iota_i(t) \parallel [x_1.u_1|x_2.u_2]{\cdot}S \rangle \rhd_{\mathrm{R}} \langle u_i[t/x_i] \parallel S \rangle$$

- *Adjoint reductions:* Function applications, projections and branching build up the context.

$$\langle t\, u \parallel S \rangle \rhd_{\mathrm{R}} \langle t \parallel u{\cdot}S \rangle \tag{1}$$
$$\langle \pi_i(u) \parallel S \rangle \rhd_{\mathrm{R}} \langle u \parallel \pi_i{\cdot}S \rangle$$
$$\langle \delta(t, x.u, y.v) \parallel S \rangle \rhd_{\mathrm{R}} \langle t \parallel [x.u|y.v]{\cdot}S \rangle \tag{2}$$

Notice that these reductions define $\rhd_{\mathrm{R}}$ as a deterministic relation: if $c \rhd_{\mathrm{R}} c'$ and $c \rhd_{\mathrm{R}} c''$ then $c' = c''$.

The reductions that we call *adjoint* are all of the form:

$$\langle f^*(t) \parallel S \rangle \rhd_{\mathrm{R}} \langle t \parallel f(S) \rangle \tag{3}$$

In other words, adjoint reductions state that the destructive operations of λ-calculus are, by analogy with linear algebra, adjoint to the constructions on contexts (Girard [26]). As a consequence, they build the context inside-out, as in our example:

$$\begin{aligned}\langle \delta((\lambda x.\iota_1(t))u, y.v, z.w) \parallel \star \rangle &\rhd_{\mathrm{R}} \langle (\lambda x.\iota_1(t))u \parallel [y.v|z.w]{\cdot}\star \rangle \\ &\rhd_{\mathrm{R}} \langle \lambda x.\iota_1(t) \parallel u{\cdot}[y.v|z.w]{\cdot}\star \rangle \\ &\rhd_{\mathrm{R}} \langle \iota_1(t[u/x]) \parallel [y.v|z.w]{\cdot}\star \rangle \\ &\rhd_{\mathrm{R}} \langle v[t[u/x]/y] \parallel \star \rangle\end{aligned}$$

In this example, the context $u{\cdot}[y.v|z.w]{\cdot}\star$ corresponds to the expression with a hole $\delta(\square\, u, y.v, z.w)$ read from the inside to the outside.

B. Solving equations for expressions

We would like to relate the evaluation of terms to their normalisation. Notice that the reductions from Figure 1c are not enough to simplify a term such as $\delta(x, y_1.\lambda z.t, y_2.\lambda z.u)\, z$ into $\delta(x, y_1.t, y_2.u)$. This requires commutation rules coming from natural deduction in logic (Figure 2). A distinct solution, as we are going to see, is to represent the various constructs of the abstract machine abstractly—as the solutions to the equations given by their transition rules. Let us explain this latter idea.

Rephrase reductions (1) and (2) as mappings from stacks to commands:

$$t\, u : S \mapsto \langle t \parallel u{\cdot}S \rangle \qquad \delta(t, x.u, y.v) : S \mapsto \langle t \parallel [x.u|y.v]{\cdot}S \rangle$$

As explained by Curien and the first author [12], one can *read* definitions off these mappings. A binder μ is introduced for the purpose:

$$t\, u \stackrel{\text{def}}{=} \mu\star.\langle t \parallel u{\cdot}\star \rangle \qquad \delta(t, x.u, y.v) \stackrel{\text{def}}{=} \mu\star.\langle t \parallel [x.u|y.v]{\cdot}\star \rangle$$

That is, *destructors are represented abstractly by their transition rule in the machine*. The expression $\mu\star.c$ maps stacks to commands thanks to the following reduction rule:

$$\langle \mu\star.c \parallel S \rangle \rhd_{\mathrm{R}} c[S/\star]$$

In fact, any equation of the form (3), assuming that f is substitutive, can be solved in this way:

$$f^*(t) \stackrel{\text{def}}{=} \mu\star.\langle t \parallel f(\star) \rangle$$

Let us pause on the idea of *solving equations* [48, 49]. This improved wording has two purposes:

- *Emphasise the conscious step taken*, to underline that abstract-machine-like calculi do not serve as replacements for λ-calculi—one still has to determine which equations are interesting to consider.
- Making us comfortable with the fact that, later in Section III, *there may be two solutions, depending on the polarity*.

$$t,u,v ::= x \mid \iota_i(t) \mid \mu\star.c \mid \mu(x\cdot\star).c \mid \mu{<}\star.c_1;\star.c_2{>} \qquad c ::= \langle t \parallel e\rangle$$
$$(e =)\ S ::= \star \mid u\cdot S \mid \pi_i\cdot S \mid [x.u|y.v]\cdot S \qquad i \in \{1,2\}$$

(a) Expressions, contexts, and commands

$$\begin{array}{rl}
\langle \mu\star.c \parallel S\rangle & \rhd_{\mathrm{R}}\ c[S/\star] \\
\langle \mu(x\cdot\star).c \parallel t\cdot S\rangle & \rhd_{\mathrm{R}}\ c[t/x, S/\star] \\
\langle \mu{<}\star.c_1;\star.c_2{>} \parallel \pi_i\cdot S\rangle & \rhd_{\mathrm{R}}\ c_i[S/\star] \\
\langle \iota_i(t) \parallel [x_1.u_1|x_2.u_2]\cdot S\rangle & \rhd_{\mathrm{R}}\ \langle u_i[t/x] \parallel S\rangle
\end{array}$$

(b) Reductions

$$\begin{array}{rl}
\lambda x.t & \stackrel{\text{def}}{=} \mu(x\cdot\star).\langle t \parallel \star\rangle \\
t\,u & \stackrel{\text{def}}{=} \mu\star.\langle t \parallel u\cdot\star\rangle \\
{<}t;u{>} & \stackrel{\text{def}}{=} \mu{<}\star.\langle t \parallel \star\rangle;\star.\langle u \parallel \star\rangle{>} \\
\pi_i(t) & \stackrel{\text{def}}{=} \mu\star.\langle t \parallel \pi_i\cdot\star\rangle \\
\delta(t, x.u, y.v) & \stackrel{\text{def}}{=} \mu\star.\langle t \parallel [x.u|y.v]\cdot\star\rangle
\end{array}$$

(c) Embedding the λ-calculus with sums

$$\Gamma \vdash t : A \quad , \quad \Gamma \mid e : A \vdash \Delta \quad , \quad \langle t \parallel e\rangle : (\Gamma \vdash \Delta) \qquad \text{where } \Delta = \star : B$$

(d) Judgements

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{}{\Gamma \mid \star : A \vdash \star : A} \qquad \frac{c : (\Gamma \vdash \star : A)}{\Gamma \vdash \mu\star.c : A} \qquad \frac{\Gamma \vdash t : A \quad \Gamma \mid e : A \vdash \Delta}{\langle t \parallel e\rangle : (\Gamma \vdash \Delta)}$$

$$\frac{\Gamma \vdash t : A_i}{\Gamma \vdash \iota_i(t) : A_1 + A_2}\ i \in \{1,2\} \qquad \frac{c : (\Gamma \vdash \star : A) \quad c' : (\Gamma \vdash \star : B)}{\Gamma \vdash \mu{<}\star.c;\star.c'{>} : A \times B} \qquad \frac{c : (\Gamma, x : A \vdash \star : B)}{\Gamma \vdash \mu(x\cdot\star).c : A \to B} \qquad \frac{\Gamma \mid S : A_i \vdash \Delta}{\Gamma \mid \pi_i\cdot S : A_1 \times A_2 \vdash \Delta}\ i \in \{1,2\}$$

$$\frac{\Gamma \mid S : C \vdash \Delta \quad \Gamma, x_i : A_i \vdash t_i : C\ (\forall i \in \{1,2\})}{\Gamma \mid [x_1.t_1|x_2.t_2]\cdot S : A_1 + A_2 \vdash \Delta} \qquad \frac{\Gamma \vdash t : A \quad \Gamma \mid S : B \vdash \Delta}{\Gamma \mid t\cdot S : A \to B \vdash \Delta}$$

(e) Typing of machines in Gentzen's sequent calculus (wrong)

Figure 3 – A first abstract-machine-like calculus with sums

The empty context $\star$ is now a context variable (or *co-variable*) bound in $\mu\star.c$ and is the only one that can be bound by μ. Originally, the notation μ comes from Parigot's $\lambda\mu$-calculus [52], where different co-variables name the different conclusions of a classical sequent. The idea of restricting to a single co-variable for modelling intuitionistic sequents with one conclusion is due to Herbelin [32].

C. Compatible reduction and its confluence

The category c is no longer restricted to occur at the top level: commands now have sub-commands. Therefore we can define the *compatible closure* $\to_{\mathrm{R}}$ of $\rhd_{\mathrm{R}}$ in an immediate way: one has $a \to_{\mathrm{R}} b$ (for a an expression, context, or command) whenever b is obtained from a by applying $\rhd_{\mathrm{R}}$ on one of its sub-commands (possibly itself).

To relate this notion of compatible closure to the one of λ-calculus we need a closure property similar to the following one:

$$\text{if } \langle t \parallel \star\rangle \to_{\mathrm{R}}^* \langle t' \parallel \star\rangle \text{ then } \langle \lambda x.t \parallel \star\rangle \to_{\mathrm{R}}^* \langle \lambda x.t' \parallel \star\rangle \qquad (4)$$

The solution is again to introduce an abstract notation for λ by solving the corresponding transition rule:

$$\begin{array}{l}
\lambda x.t : u\cdot S \mapsto \langle t[u/x] \parallel S\rangle \\
\lambda x.t \stackrel{\text{def}}{=} \mu(x\cdot\star).\langle t \parallel \star\rangle \qquad (\star \notin \mathbf{fv}(t)) \qquad (5)
\end{array}$$

Thus we extend μ to match patterns on stacks:

$$\langle \mu(x\cdot\star).c \parallel t\cdot S\rangle \rhd_{\mathrm{R}} c[t/x, S/\star]$$

Thanks to the abstract definition of λ, our streamlined definition of compatible closure implies the clause (4):

$$\text{if } \langle t \parallel \star\rangle \to_{\mathrm{R}}^* \langle t' \parallel \star\rangle \text{ then } \mu(x\cdot\star).\langle t \parallel \star\rangle \to_{\mathrm{R}}^* \mu(x\cdot\star).\langle t' \parallel \star\rangle$$

For the same reason, pairs ${<}t;u{>}$ are represented abstractly as a pair of transition rules:

$${<}t;u{>} \quad : \quad \pi_1\cdot S \mapsto \langle t \parallel S\rangle \quad , \quad \pi_2\cdot S \mapsto \langle u \parallel S\rangle$$

$${<}t;u{>} \stackrel{\text{def}}{=} \mu{<}\star.\langle t \parallel \star\rangle;\star.\langle u \parallel \star\rangle{>}$$

We summarise in Figure 3 the calculus that we obtain so far by adding μ to the abstract machine and by removing the constructs that can be defined. It is easy to notice that one has for all t, e, c: $\star \in \mathbf{fv}(e)$ and $\star \in \mathbf{fv}(c)$ but $\star \notin \mathbf{fv}(t)$. In particular the proviso in (5) is always satisfied.

Thanks to the separation between expressions, contexts and commands, *we have a reduction $\rhd_{\mathrm{R}}$ which is left-linear and has no critical pair*. As a consequence, $\to_{\mathrm{R}}$ *is confluent*, by application of Tait and Martin-Löf's parallel reduction technique (see for instance Nipkow [51]). This argument will hold for all the abstract-machine-like calculi of the article.

D. Typing of machines

The typing of abstract-machine-like calculi in Gentzen's sequent calculus is standard. (See Wadler [66], Curien and M.-M. [12], for introductions.)

We introduce the type system for the calculus considered so far in Figure 3e. The correspondence with sequent calculus is not obtained yet:

1) Left-introduction rules cannot be performed on arbitrary formulae; they stack.
2) The rule for $[x.u \mid y.v]\cdot S$ is still the one from natural deduction.

E. Solving equations for contexts

The calculus in Figure 3 still fails to normalize terms without using commutations, for instance in the following:

$$\langle \delta(x, y_1.\lambda z.t, y_2.\lambda z.u)\, z \parallel \star\rangle \rhd_{\mathrm{R}}^* \langle x \parallel [y_1.\lambda z.t|y_2.\lambda z.t]\cdot z\cdot\star\rangle$$

The commutation on expressions is only rephrased as a similar commutation on contexts.

$\mathbf{L}^{\ominus}_{\text{int}}$ consists in Figure 3 extended as follows:

$$S ::= \ldots \mid \tilde{\mu}[x.c|y.c'] \qquad e ::= S \mid \tilde{\mu}x.c$$

(a) Stacks and contexts

$$\langle t \parallel \tilde{\mu}x.c\rangle \rhd_{\mathrm{R}} c[t/x]$$
$$\langle \iota_i(t) \parallel \tilde{\mu}[x_1.c_1|x_2.c_2]\rangle \rhd_{\mathrm{R}} c_i[t/x_i]$$

(b) New reductions

$[x.t|y.u]\cdot S$ is removed from the grammar

$$[x.t|y.u]\cdot\star \stackrel{\text{def}}{=} \tilde{\mu}\big[x.\langle u \parallel \star\rangle \big| y.\langle v \parallel \star\rangle\big]$$

(c) Definition of branching

$$\frac{c : (\Gamma, x : A \vdash \Delta)}{\Gamma \mid \tilde{\mu}x.c : A \vdash \Delta} \qquad \frac{c : (\Gamma, x : A \vdash \Delta) \quad c' : (\Gamma, y : B \vdash \Delta)}{\Gamma \mid \tilde{\mu}[x.c|y.c'] : A + B \vdash \Delta}$$

(d) New rules

$$\frac{\Gamma \vdash t : A \quad \Gamma \mid e : B \vdash \Delta}{\Gamma \mid t\cdot e : A \to B \vdash \Delta} \qquad \frac{\Gamma \mid e : A \vdash \Delta}{\Gamma \mid \pi_1\cdot e : A \times B \vdash \Delta}$$

(e) Derivable rules (modulo weakening)

Figure 4 – The calculus $\mathbf{L}^{\ominus}_{\text{int}}$

Once again, let us describe the context $[x.u|y.v]\cdot S$ as a pair of mappings:

$$\iota_1(x) \mapsto \langle u \parallel S\rangle \qquad , \qquad \iota_2(y) \mapsto \langle v \parallel S\rangle$$

This suggests that the branching context can be written abstractly, by introducing a new binder $\tilde{\mu}$:

$$[x.u|y.v]\cdot S = \tilde{\mu}\big[x.\langle u \parallel S\rangle \big| y.\langle v \parallel S\rangle\big] \tag{6}$$

The stack $\tilde{\mu}[x_1.c_1|x_2.c_2]$ is symmetric in shape to the constructor for pairs, and associates to any injection ι_i the appropriate command c_1 or c_2:

$$\langle \iota_i(t) \parallel \tilde{\mu}[x_1.c_1|x_2.c_2]\rangle \rhd_{\mathrm{R}} c_i[t/x_i] \qquad (\forall i \in \{1,2\})$$

In fact, we only consider the contexts $[x.u\,|\,y.v]\cdot\star$ in the definition of branching:

$$\delta(t, x.u, y.v) \stackrel{\text{def}}{=} \mu\star.\big\langle t \,\big\|\, \tilde{\mu}\big[x.\langle u \parallel \star\rangle \big| y.\langle v \parallel \star\rangle\big]\big\rangle \tag{7}$$

While S was duplicated in (6), this definition is local.

We may as well consider the context $\tilde{\mu}x.c$ which can be used to represent an arbitrary mapping from expressions to commands:

$$\langle t \parallel \tilde{\mu}x.c\rangle \rhd_{\mathrm{R}} c[t/x]$$

It is easy to see that a context $\tilde{\mu}x.\langle u \parallel \star\rangle$ behaves like the expression with a hole $(\lambda x.u)\,\square$. Now, the existence of the command $\langle \mu\star.c \parallel \tilde{\mu}x.c'\rangle$ shows that we cannot consider the context $\tilde{\mu}x.c$ among the grammar of stacks S, since we do not want to introduce a critical pair in the reduction $\rhd_{\mathrm{R}}$. Therefore we now make contexts a strict superset of stacks S—we arrive at Curien and Herbelin's characterisation of call by name [11].

Notice that $\tilde{\mu}[x.c\,|\,y.c']$ belongs to the syntactic category S of stacks. (It no longer describes stacks *per se*, but any context against which the μ reduces.) We may criticise the call-by-name bias for the first time, for introducing this arbitrary difference between $\tilde{\mu}x.c$ and $\tilde{\mu}[x_1.c_1|x_2.c_2]$, while we motivated both in the same way.

The calculus $\mathbf{L}^{\ominus}_{\text{int}}$ (Figure 4) summarises our development so far. Since $\rhd_{\mathrm{R}}$ is still without critical pairs, $\to_{\mathrm{R}}$ is still confluent.

F. Commutation rules are redundant

The term $\delta(x, y_1.\lambda z.t, y_2.\lambda z.u)\,z$ now reduces as follows:

$$\begin{aligned}
&\langle \delta(x, y_1.\lambda z.t, y_2.\lambda z.u)\,z \parallel S\rangle \\
&\rhd_{\mathrm{R}} \langle \delta(x, y_1.\lambda z.t, y_2.\lambda z.u) \parallel z\cdot S\rangle \\
&\rhd_{\mathrm{R}} \big\langle x \,\big\|\, \tilde{\mu}\big[y_1.\langle \lambda z.t \parallel z\cdot S\rangle \big| y_2.\langle \lambda z.u \parallel z\cdot S\rangle\big]\big\rangle \\
&\to^*_{\mathrm{R}} \big\langle x \,\big\|\, \tilde{\mu}\big[y_1.\langle t \parallel S\rangle \big| y_2.\langle u \parallel S\rangle\big]\big\rangle
\end{aligned}$$

In particular, $\langle \delta(x, y_1.\lambda z.t, y_2.\lambda z.u)\,z \parallel \star\rangle$ does not reduce to $\langle \delta(x, y_1.t, y_2.u) \parallel \star\rangle$ despite what commutation rules would prescribe. However, they have the following common reduct:

$$\big\langle x \,\big\|\, \tilde{\mu}\big[y_1.\langle t \parallel \star\rangle \big| y_2.\langle u \parallel \star\rangle\big]\big\rangle$$

The same happens with the two other commutation rules. In other words, *commutation rules are redundant in* $\boldsymbol{L}^{\ominus}_{int}$, *and at the same time* $\boldsymbol{L}^{\ominus}_{int}$ *offers a novel reduction theory compared to the* λ*-calculus with sums and commutations*. In particular, normalisation in the proof theoretic sense is obtained with the compatible closure ($\to_{\mathrm{R}}$) of evaluation ($\rhd_{\mathrm{R}}$).

G. Focalisation

Any proof in the propositional intuitionistic sequent calculus can be retrieved as an $\mathbf{L}^{\ominus}_{\text{int}}$ derivation. Indeed, the typing rule for $\tilde{\mu}$ (Figure 4d) allows left-introduction rules to be performed on arbitrary formulae—in particular, contexts $u\cdot e$ and $\pi_i\cdot e$ corresponding to unrestricted left-introduction rules (Figure 4e) can be defined, using $\tilde{\mu}$, as the solutions to the following equations:

$$\langle t \parallel u\cdot e\rangle \rhd_{\mathrm{R}} \big\langle \mu\star.\langle t \parallel u\cdot\star\rangle \,\big\|\, e\big\rangle$$
$$\langle t \parallel \pi_i\cdot e\rangle \rhd_{\mathrm{R}} \big\langle \mu\star.\langle t \parallel \pi_i\cdot\star\rangle \,\big\|\, e\big\rangle$$

often called *ς-reductions*. The rules from Figure 4e are derivable modulo weakening of the premiss e, which is admissible in the standard way. (For simplicity we did not formulate $\mathbf{L}^{\ominus}_{\text{int}}$ in the multiplicative style with explicit weakening, which would let us state directly that the unrestricted rules are derivable.)

In words, *unrestricted left-introduction rules in* $\boldsymbol{L}^{\ominus}_{int}$ *hide cuts*. We call *focalisation* this phenomenon by which certain introduction rules hide cuts. It is indeed responsible for the shape of synchronous phases in focusing.

H. Inside-out contexts are primitive

Every context e corresponds to a term with a hole $\mathrm{E}_e[\,]$. Thus, every command $\langle t \parallel e\rangle$ (or equivalently every expression $\mu\star.\langle t \parallel e\rangle$) corresponds to a λ-term $\mathrm{E}_e[t]$. We map commands to expressions as follows:

$$\mathrm{E}_{\star}\square \stackrel{\text{def}}{=} \square \qquad \mathrm{E}_{\pi_i\cdot S}\square \stackrel{\text{def}}{=} \mathrm{E}_S[\pi_i(\square)]$$
$$\mathrm{E}_{\tilde{\mu}x.\langle t \parallel e\rangle}\square \stackrel{\text{def}}{=} (\lambda x.\mathrm{E}_e[t])\square \qquad \mathrm{E}_{V\cdot S}\square \stackrel{\text{def}}{=} \mathrm{E}_S[\square\, V]$$
$$\mathrm{E}_{\tilde{\mu}[x.\langle t \parallel e\rangle | y.\langle u \parallel e'\rangle]}\square \stackrel{\text{def}}{=} \delta(\square, x.\mathrm{E}_e[t], y.\mathrm{E}_{e'}[u])$$

In words, a command $\langle t \parallel e\rangle$ is converted into an expression by rewinding the adjoint reductions leading to e and by expressing $\tilde{\mu}$ as λ or δ. The definition ensures that $\square$ appears exactly once in $\mathrm{E}_e[\,]$—indeed an expression with a hole.

Conversely, in the presence of $\tilde{\mu}$, any expression with a hole $E[\,]$ can be represented as $\tilde{\mu}x.\langle E[x] \parallel \star\rangle$. Furthermore, the latter gets, after reduction, the canonical form that we expect. For instance, considering $E_1[\,] \stackrel{\text{def}}{=} (\lambda x.t)\square$ and $E_2[\,] \stackrel{\text{def}}{=} \delta(\square t, y.u, z.v)$, we have:

$$\tilde{\mu}x.\langle E_1[x] \parallel \star\rangle \to^*_{\mathrm{R}} \tilde{\mu}x.\langle t \parallel \star\rangle$$
$$\tilde{\mu}x.\langle E_2[x] \parallel \star\rangle \to^*_{\mathrm{R}} \tilde{\mu}x.\big\langle x \,\big\|\, t\cdot\tilde{\mu}\big[y.\langle u \parallel \star\rangle \big| z.\langle v \parallel \star\rangle\big]\big\rangle$$

where $\tilde{\mu}x.\langle t \parallel \star\rangle$ and $t{\cdot}\tilde{\mu}\big[y.\langle u \parallel \star\rangle \big| z.\langle v \parallel \star\rangle\big]$ are the contexts that we can expect to obtain from $(\lambda x.t)\,\square$ and $\delta(\square\, t, y.u, z.v)$. In fact, provided that we refine the definition of abstraction as $\lambda x.t \stackrel{\text{def}}{=} \mu(y{\cdot}\star).\,\big\langle\, y \,\big\|\, \tilde{\mu}x.\langle t \parallel \star\rangle \,\big\rangle$ (as implied in Espírito Santo [18]), one has for any context e:

$$\langle \mathrm{E}_e[t] \parallel \star\rangle \to_{\mathrm{R}}^{*} \langle t \parallel e\rangle$$

(It might matter in other contexts that the reduction is furthermore a single *parallel reduction* step.)

In this sense, contexts are equivalent to expressions with a hole. But we argue that *inside-out contexts are more primitive*.

- When the reduction is directly defined on expressions $E[t]$, an external property—the so-called unique context lemma—replaces the adjoint reductions in the role of reaching a main reduction.
- It is more difficult to express inside-out contexts as expressions with a hole than the contrary, as we have seen above, because it requires an external definition by induction.

Inside-out contexts reveal intermediate steps in the reduction, which, once they are expressed as part of the formalism, render such external properties and definitions unnecessary.

III. Polarisation and Extensional Sums

A. Introducing expansions

1) Conversions on expressions vs. on commands: In this section, we consider extensionality principles formulated as expansions $\rhd_{\mathrm{E}}$ defined, unlike $\rhd_{\mathrm{R}}$, on expressions and on contexts. The following expansions are standard:

$$t \rhd_{\mathrm{E}} \mu\star.\langle t \parallel \star\rangle \qquad , \qquad e \rhd_{\mathrm{E}} \tilde{\mu}x.\langle x \parallel e\rangle \quad (x \notin \mathbf{fv}(e))$$

They enunciate that one has $t \simeq_{\mathrm{RE}} u$ if and only if for all e, $\langle t \| e\rangle \simeq_{\mathrm{RE}} \langle u \| e\rangle$ (and symmetrically for contexts): there is only one notion of equivalence which coincides between expressions, contexts and commands.

2) Extensionality for sums with a regular shape: Converting, as described in Section II-H, contexts e into expressions with a hole $E[\,]$ (and conversely) shows that the following:

$$e \simeq_{\mathrm{RE}} \tilde{\mu}\big[x.\langle \iota_1(x) \parallel e\rangle \big| y.\langle \iota_2(y) \parallel e\rangle\big]$$

is equivalent to the standard η-expansion of sums:

$$E[t] \simeq_{\mathrm{RE}} \delta(t, x.E[\iota_1(x)], y.E[\iota_2(y)])\,. \tag{8}$$

In fact, the traditional η-expansions of sequent calculus suggests extensionality axioms that all have a regular shape:

$$\begin{aligned} t &\rhd_{\mathrm{E}} \mu(x{\cdot}\star).\langle t \parallel x{\cdot}\star\rangle \\ t &\rhd_{\mathrm{E}} \mu{<}\star.\langle t \parallel \pi_1{\cdot}\star\rangle; \star.\langle t \parallel \pi_2{\cdot}\star\rangle{>} \\ e &\rhd_{\mathrm{E}} \tilde{\mu}\big[x.\langle \iota_1(x) \parallel e\rangle \big| y.\langle \iota_2(y) \parallel e\rangle\big] \end{aligned} \tag{9}$$

However, the degeneracy of λ-calculus with extensional sums and fixed points mentioned in the introduction applies: in untyped $\mathbf{L}_{\mathrm{int}}^{\ominus}$, one has $t \simeq_{\mathrm{RE}} u$ for any two expressions t and u, by an immediate adaptation of Dougherty [16].

3) Positive sums: Positive sums, unlike the sums of $\mathbf{L}_{\mathrm{int}}^{\ominus}$, are called by value. A difference is therefore introduced, among expressions, between *values* V (such as $\iota_1(x)$) and non-values (such as $t\,u$), as follows:

- The reduction of $\tilde{\mu}$ is restricted to values:
$$\langle V \parallel \tilde{\mu}x.c\rangle \rhd_{\mathrm{R}} c[V/x]$$
- The reduction of a non-value $\mu\star.c$ (including $t\,u$) takes precedence over the one defined by the context; in other words every positive context is a stack.

As a consequence, the equation (9) no longer converts an arbitrary context into a stack. In addition, it becomes equivalent to the following η-expansion for sums restricted to values:

$$E[V] \simeq_{\mathrm{RE}} \delta(V, x.E[\iota_1(x)], y.E[\iota_2(y)])$$

Thus, when sums are positive, (as observed for the classical sequent calculus [13],) the extensionality axiom does not interfere with the order of evaluation.

B. The polarised calculus $\boldsymbol{L}_{int}$

$\mathbf{L}_{\mathrm{int}}$ is a polarised variant of $\mathbf{L}_{\mathrm{int}}^{\ominus}$ introduced in Figure 5. In addition to the negative connectives $\to, \times$ it proposes positive pairs and sums $(\otimes, \oplus)$.

1) Polarisation as a locally-determined strategy: Polarisation, unlike the call-by-value evaluation strategy, assigns a strict evaluation to sums without changing the evaluation order of the other connectives of $\mathbf{L}_{\mathrm{int}}^{\ominus}$. The evaluation order is determined at the level of each command by a polarity $\varepsilon \in \{\ominus, +\}$ assigned to every expression and every context, impacting the reduction rules as follows:

Negative polarity	Every expression is a value (CBN)
Positive polarity	Every context is a stack (CBV)

Polarities in this sense are similar to Danos, Joinet and Schellinx's [13] "t/q" annotations on the classical sequent calculus—$\mathbf{L}_{\mathrm{int}}$ in fact determines an intuitionistic variant of their polarised classical logic LK_p^{η}.

2) The subscript/superscript convention: The polarity of an expression ($t_{\ominus}$ or t_{+}) or a context ($e_{\ominus}$ or e_{+}) is defined in the Figures 5c and 5d. Whenever necessary, the polarity annotation is part of the grammar: that is variables ($x^{\ominus}$, x^{+}), binders ($\mu^{\ominus}\star.c$, $\mu^{+}\star.c$), and type variables ($X^{\ominus}$, X^{+}). These notations conform to the following convention for polarity annotations:

- a polarity in superscript (x^{ϵ}) indicates an annotation which is part of the grammar,
- a polarity in subscript (t_{ε}) is not part of the grammar and only asserts that the term has this polarity.

3) Polarisation as an approximation of types: A system of simple types, which corresponds to an intuitionistic sequent calculus, is provided on top of the Curry-style $\mathbf{L}_{\mathrm{int}}$. It is easy to see that:

- if one has $\Gamma \vdash t : A_{\varepsilon}$ then t has polarity ε,
- if one has $\Gamma \mid e : A_{\varepsilon} \vdash \Delta$ then e has polarity ε.

Therefore, for typeable terms, the type determines the order of evaluation.

Calculating on terms together with their typing derivations is tedious—it requires intimate knowledge of the properties of the type system (for instance subject reduction). With more complex type systems, such properties are even uncertain.

What we show with $\mathbf{L}_{\mathrm{int}}$ is that all the information relevant for evaluation can be captured in a grammar, requiring no complete inspection of the terms, thanks to appropriate polarity annotations. We only needed to make sure that the conversion rules are grammatically well defined despite the presence of annotations, but this is easy to verify.

On the other hand, polarities are not types:

- like in the pure λ calculus, it is possible to define arbitrary fixed-points, and in particular $\mathbf{L}_{\mathrm{int}}$ is Turing-complete.
- it is possible to obtain ill-formed commands $\langle t_{\ominus} \parallel e_{+}\rangle$ by reducing well-polarised (but ill-typed) commands such as $\big\langle \mu(x^{\ominus}{\cdot}\star).\langle x^{\ominus} \parallel \star\rangle \,\big\|\, t_{\ominus}{\cdot}e_{+} \big\rangle$. Such mixed commands do not

… : Main additions to Figure 4 Refer to the subscript/superscript convention in Section III-B2

$$\varepsilon ::= \ominus \mid +$$

(a) Polarities

$$c ::= \langle t \parallel e \rangle$$

(b) Commands

$$t,u\begin{cases} t_\ominus, u_\ominus ::= x^\ominus \mid \mu^\ominus\star.c \mid \mu(x^\varepsilon\cdot\star).c \mid \mu{<}\star.c;\star.c'{>} \\ t_+, u_+ ::= x^+ \mid (V,W) \mid \iota_i(V) \mid \mu^+\star.c \\ V, W ::= x^+ \mid (V,W) \mid \iota_i(V) \mid t_\ominus \end{cases}$$

(c) Expressions and values

$$e\begin{cases} e_\ominus ::= \star \mid V\cdot S \mid \pi_i\cdot S \mid \tilde\mu x^\ominus.c \\ e_+ ::= \star \mid \tilde\mu x^+.c \mid \tilde\mu(x^{\varepsilon_1}, y^{\varepsilon_2}).c \mid \tilde\mu[x^{\varepsilon_1}.c \mid y^{\varepsilon_2}.c] \\ S ::= \star \mid V\cdot S \mid \pi_i\cdot S \mid e_+ \end{cases}$$

(d) Contexts and stacks

$$\begin{array}{lcll} (R\tilde\mu) & \langle V_\varepsilon \parallel \tilde\mu x^\varepsilon.c\rangle & \rhd_{\mathrm{R}} & c[V_\varepsilon/x^\varepsilon] \\ (R\mu) & \langle \mu^\varepsilon\star.c \parallel S\rangle & \rhd_{\mathrm{R}} & c[S/\star] \\ (R\to) & \langle \mu(x^\varepsilon\cdot\star).c \parallel V_\varepsilon\cdot S\rangle & \rhd_{\mathrm{R}} & c[V_\varepsilon/x^\varepsilon, S/\star] \\ (R\times_i) & \langle \mu{<}\star.c_1;\star.c_2{>} \parallel \pi_i\cdot S\rangle & \rhd_{\mathrm{R}} & c_i[S/\star] \\ (R\otimes) & \langle (V_{\varepsilon_1}, W_{\varepsilon_2}) \parallel \tilde\mu(x^{\varepsilon_1}, y^{\varepsilon_2}).c\rangle & \rhd_{\mathrm{R}} & c[V_{\varepsilon_1}/x^{\varepsilon_1}, W_{\varepsilon_2}/y^{\varepsilon_2}] \\ (R\oplus_i) & \langle \iota_i(V_{\varepsilon_i}) \parallel \tilde\mu[x_1^{\varepsilon_1}.c_1 \mid x_2^{\varepsilon_2}.c_2]\rangle & \rhd_{\mathrm{R}} & c_i[V_{\varepsilon_i}/x_i^{\varepsilon_i}] \end{array}$$

(e) Reductions[1]

$$\begin{array}{lcll} (E\mu) & t_\varepsilon & \rhd_{\mathrm{E}} & \mu^\varepsilon\star.\langle t_\varepsilon \parallel \star\rangle \\ (E\to) & t_\ominus & \rhd_{\mathrm{E}} & \mu(x\cdot\star).\langle t_\ominus \parallel x\cdot\star\rangle \\ (E\times) & t_\ominus & \rhd_{\mathrm{E}} & \mu{<}\star.\langle t_\ominus \parallel \pi_1\cdot\star\rangle;\star.\langle t_\ominus \parallel \pi_2\cdot\star\rangle{>} \\ (E\tilde\mu) & e_\varepsilon & \rhd_{\mathrm{E}} & \tilde\mu x^\varepsilon.\langle x^\varepsilon \parallel e_\varepsilon\rangle \\ (E\otimes) & e_+ & \rhd_{\mathrm{E}} & \tilde\mu(x,y).\langle (x,y) \parallel e_+\rangle \\ (E\oplus) & e_+ & \rhd_{\mathrm{E}} & \tilde\mu\big[x.\langle \iota_1(x) \parallel e_+\rangle \big| y.\langle \iota_2(y) \parallel e_+\rangle\big] \end{array}$$

(f) Expansions

$$A,B\begin{cases} A_\ominus, B_\ominus ::= X^\ominus \mid A\to B \mid A\times B \\ A_+, B_+ ::= X^+ \mid A\otimes B \mid A\oplus B \end{cases}$$

$x : A$ stands for $x^\varepsilon : A_\varepsilon$

(g) Types

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{}{\Gamma \mid \star : A \vdash \star : A} \qquad \frac{c : (\Gamma, x : A \vdash \Delta)}{\Gamma \mid \tilde\mu x.c : A \vdash \Delta} \qquad \frac{c : (\Gamma \vdash \star : A_\varepsilon)}{\Gamma \vdash \mu^\varepsilon\star.c : A_\varepsilon} \qquad \frac{\Gamma \vdash t : A \quad \Gamma \mid e : A \vdash \Delta}{\langle t \parallel e\rangle : (\Gamma \vdash \Delta)}$$

(h) Identity

$$\frac{c : (\Gamma, x : A \vdash \star : B)}{\Gamma \vdash \mu(x\cdot\star).c : A\to B} \qquad \frac{c_i : (\Gamma \vdash \star : A_i) \quad (\forall i\in\{1,2\})}{\Gamma \vdash \mu{<}\star.c_1;\star.c_2{>} : A_1\times A_2} \qquad \frac{\Gamma \vdash V : A \quad \Gamma \vdash W : B}{\Gamma \vdash (V,W) : A\otimes B} \qquad \frac{\Gamma \vdash V : A_i}{\Gamma \vdash \iota_i(V) : A_1\oplus A_2}$$

$$\frac{\Gamma \vdash V : A \quad \Gamma \mid S : B \vdash \Delta}{\Gamma \mid V\cdot S : A\to B \vdash \Delta} \qquad \frac{\Gamma \mid S : A_i \vdash \Delta}{\Gamma \mid \pi_i\cdot S : A_1\times A_2 \vdash \Delta} \qquad \frac{c : (\Gamma, x : A, y : B \vdash \Delta)}{\Gamma \mid \tilde\mu(x,y).c : A\otimes B \vdash \Delta} \qquad \frac{c_i : (\Gamma, x_i : A_i \vdash \Delta) \quad (\forall i\in\{1,2\})}{\Gamma \mid \tilde\mu[x_1.c_1 \mid x_2.c_2] : A_1\oplus A_2 \vdash \Delta}$$

(i) Logic

Figure 5 – $\mathbf{L}_{\mathrm{int}}$: grammar (top) — conversions (middle) — simple types (bottom)

reduce, and the reader should not expect to see them appear in the rest of the article.

4) Confluence: Notice that an expression is never both positive and negative, and only the empty context ($\star$) is both positive and negative. In particular, $\mu^+\star.c$ is never a value, and $\tilde\mu x^\ominus.c$ is never a stack. With this observation in mind, it is easy to see that once again $\rhd_{\mathrm{R}}$ has no critical pairs. Therefore $\to_{\mathrm{R}}$ is confluent.

$\to_{\mathrm{RE}}$ is not confluent given the possibility of ill-typed expansions, and we only conjecture the consistency of $\simeq_{\mathrm{RE}}$. It is already known that the extensional λ-calculus with surjective pairs is consistent, with interesting proofs from denotational semantics (Lambek and Scott [36]) and rewriting (Støvring [61]). Extending either technique is interesting, and one reason why this has not been done before can be that relaxing the associativity of composition might not seem natural at first.

5) Adjoint equations have two solutions: Since there are now two distinct binders $\mu^\ominus\star.c$ and $\mu^+\star.c$, adjoint equations $\langle f^*(t) \parallel S\rangle \rhd_{\mathrm{R}} \langle t \parallel f(S)\rangle$ for substitutive f now have two solutions depending on the polarity of S:

$$f^*(t)^\varepsilon \stackrel{\text{def}}{=} \mu^\varepsilon\star.\langle t \parallel f(\star)\rangle$$

For instance there is actually one version of branching $\delta(t, x.u, y.v)^\varepsilon$ for each $\varepsilon \in \{\ominus, +\}$, only one of which is a value (its evaluation is delayed), and similarly for function application: $(t\,u)^\varepsilon$.

This splitting reflects the fact that according to polarisation, $\delta(t, x.u, y.v)$ is either strict or lazy depending on whether u and v are strict or lazy. This observation has several implications studied elsewhere:

- A λ-calculus (or natural deduction) based on $\mathbf{L}_{\mathrm{int}}$ would define the polarity of an expression, such as a case or a let-binder, by a shallow inspection. Function application would require an explicit annotation that determines the polarity of the result; for a programming language, this information would be synthesised during type inference. (See [49], though in the context of classical logic.)
- One should not seek to model $\mathbf{L}_{\mathrm{int}}$ in a category directly: composition is not associative, but defines a pre-duploid [50]. The pre-duploid becomes a duploid once *shifts* $\Downarrow \dashv \Uparrow$ are added to $\mathbf{L}_{\mathrm{int}}$. We omitted such shifts in this article for

[1] The rule $(R\mu)$ appeared in print as $\langle \mu^\varepsilon\star.c \parallel S_\varepsilon\rangle \rhd_{\mathrm{R}} c[S_\varepsilon/\star]$ (with the extra polarity constraint on S). This extra constraint was incorrect, as it invalidated the closure under substitutions $[S/\star]$. *(Added December 2015.)*

$$
\begin{aligned}
x^A &\stackrel{\text{def}}{=} x^\varepsilon \\
\lambda x^A.t &\stackrel{\text{def}}{=} \mu(y^\varepsilon\cdot\star).\langle y^\varepsilon \parallel \tilde{\mu}x^\varepsilon.\langle t \parallel \star\rangle\rangle \\
(t\,u^B)^A &\stackrel{\text{def}}{=} \mu^\varepsilon\star.\langle u \parallel \tilde{\mu}x^{\varepsilon_2}.\langle t \parallel x^{\varepsilon_2}\cdot\star\rangle\rangle \\
<t;u> &\stackrel{\text{def}}{=} \mu<\star.\langle t \parallel \star\rangle;\star.\langle u \parallel \star\rangle> \\
\pi_i(t)^A &\stackrel{\text{def}}{=} \mu^\varepsilon\star.\langle t \parallel \pi_i\cdot\star\rangle \\
\iota_i(t^A) &\stackrel{\text{def}^\dagger}{=} \mu^+\star.\langle t \parallel \tilde{\mu}x^+.\langle \iota_i(x^+) \parallel \star\rangle\rangle \\
\delta(t,x.u,y.v)^A &\stackrel{\text{def}}{=} \mu^\varepsilon\star.\langle t \parallel \tilde{\mu}\left[x.\langle u \parallel \star\rangle \middle| y.\langle v \parallel \star\rangle\right]\rangle
\end{aligned}
$$

where A has polarity ε and B has polarity ε_2.
†: when t^A is not a value.

(a) Definition in $\mathbf{L}^{\vdash}_{\text{int}}$ of the operations of $\Lambda^{\vdash}$, inducing a translation $\mathrm{S}[\![\cdot]\!] : \Lambda^{\vdash} \to \mathbf{L}^{\vdash}_{\text{int}}$.

$$
\begin{aligned}
\mathrm{N}[\![x^A]\!] &\stackrel{\text{def}}{=} x^A \\
\mathrm{N}[\![\mu^A\star.c]\!] &\stackrel{\text{def}}{=} \mathrm{N}[\![c]\!] \\
\mathrm{N}[\![\mu(x^A\cdot\star).c]\!] &\stackrel{\text{def}}{=} \lambda x^A.\mathrm{N}[\![c]\!] \\
\mathrm{N}[\![\mu<\star.c;\star.c'>]\!] &\stackrel{\text{def}}{=} <\mathrm{N}[\![c]\!];\mathrm{N}[\![c']\!]> \\
\mathrm{N}[\![\iota_1(V)]\!] &\stackrel{\text{def}}{=} \iota_i(\mathrm{N}[\![V]\!]) \\
\mathrm{N}[\![\langle t \parallel e\rangle]\!] &\stackrel{\text{def}}{=} \mathrm{E}_e[\mathrm{N}[\![t]\!]] \\
\mathrm{E}_\star\Box &\stackrel{\text{def}}{=} \Box \\
\mathrm{E}_{\pi_i\cdot S}\Box &\stackrel{\text{def}}{=} \mathrm{E}_S[\pi_i(\Box)] \\
\mathrm{E}_{V\cdot S}\Box &\stackrel{\text{def}}{=} \mathrm{E}_S[\Box\,\mathrm{N}[\![V]\!]] \\
\mathrm{E}_{\tilde{\mu}x^A.c}\Box &\stackrel{\text{def}}{=} (\lambda x^A.\mathrm{N}[\![c]\!])\Box \\
\mathrm{E}_{\tilde{\mu}[x^A.c\,|\,y^B.c']}\Box &\stackrel{\text{def}}{=} \delta(\Box, x^A.\mathrm{N}[\![c]\!], y^B.\mathrm{N}[\![c']\!])
\end{aligned}
$$

(b) Translation $\mathrm{N}[\![\cdot]\!] : \mathbf{L}^{\vdash}_{\text{int}} \to \Lambda^{\vdash}$

Figure 6 – Translations between $\Lambda^{\vdash}$ and $\mathbf{L}^{\vdash}_{\text{int}}$

simplicity, they can be added exactly as in [50].

C. Relating $\boldsymbol{L}_{int}$ to the λ-calculus with sums

We relate typed restrictions of $\mathbf{L}_{\text{int}}$ (minus the positive pair $\otimes$) and the λ-calculus with sums. Let us denote with $\mathbf{L}^{\vdash}_{\text{int}}$ and $\Lambda^{\vdash}$ the simply-typed $\mathbf{L}_{\text{int}}$ (without $\otimes$) and λ-calculus with sums, that is to say restricted by the typing judgements from Figures 5 (minus the positive pair $\otimes$) and 1 respectively. For simplicity we will omit the typing judgements and may choose to write explicit type annotations (*à la* Church). We give these systems the same grammar of types by assuming that the set of type variables X of $\Lambda^{\vdash}$ coincides with the set of positive and negative type variables $X^+, X^\ominus$ of $\mathbf{L}^{\vdash}_{\text{int}}$ and by identifying the connectives $+$ and $\oplus$. In Figure 6 we define two translations:

$$
\mathrm{N}[\![\cdot]\!] : \mathbf{L}^{\vdash}_{\text{int}} \to \Lambda^{\vdash} \qquad , \qquad \mathrm{S}[\![\cdot]\!] : \Lambda^{\vdash} \to \mathbf{L}^{\vdash}_{\text{int}}
$$

These translations are well defined: if $\Gamma \vdash_{\mathbf{L}_{\text{int}}} t : A$ then $\Gamma \vdash_{\Lambda} \mathrm{N}[\![t]\!] : A$ and if $\Gamma \vdash_{\Lambda} t : A$ then $\Gamma \vdash_{\mathbf{L}_{\text{int}}} \mathrm{S}[\![t]\!] : A$. In addition:

Proposition 1. *1) If* $\Gamma \vdash_{\boldsymbol{L}_{int}} t : A$ *then* $t \simeq_{\text{RE}} \mathrm{S}[\![\mathrm{N}[\![t]\!]]\!]$.
2) If $\Gamma \vdash_{\Lambda} t : A$ *then* $t \approx \mathrm{N}[\![\mathrm{S}[\![t]\!]]\!]$.
3) If $t \simeq_{\text{RE}} u$ *in* $\boldsymbol{L}^{\vdash}_{int}$ *then* $\mathrm{N}[\![t]\!] \approx \mathrm{N}[\![u]\!]$.

Of course, it cannot be the case that $t \approx u$ in $\Lambda^{\vdash}$ implies $\mathrm{S}[\![t]\!] \simeq_{\text{RE}} \mathrm{S}[\![u]\!]$ in $\mathbf{L}^{\vdash}_{\text{int}}$, unless $\mathbf{L}_{\text{int}}$ identifies all expressions. Indeed, one would have $(\lambda x^+.(b^\ominus\,\lambda a.x^+))(y\,z)^+ \simeq_{\text{RE}} b^\ominus\,\lambda a.(y\,z)^+$ in $\mathbf{L}_{\text{int}}$. For $y = \lambda z.t_+$ and $b_\ominus = \lambda z^\ominus.\mu^\ominus\star.c[(z_\ominus\,a)^+/^f x^+]$ when t_+ and c are arbitrary, this implies $\langle t_+ \parallel \tilde{\mu}x^+.c\rangle \simeq_{\text{RE}} c[t_+/^f x^+]$—where the *focalising substitution* $[u_+/^f x^+]$ is defined using focalisation in the sense of Section II-G:

Definition 2. Unrestricted expressions and contexts are defined as follows (in addition to $\iota_i(t_+)$ defined as in Figure 6a):

$$
\begin{aligned}
t_+\cdot e &\stackrel{\text{def}^\dagger}{=} \tilde{\mu}x^\ominus.\langle t_+ \parallel \tilde{\mu}y^+.\langle x^\ominus \parallel y^+\cdot e\rangle\rangle \\
V\cdot e_\ominus &\stackrel{\text{def}^\ddagger}{=} \tilde{\mu}x^\ominus.\langle \mu^\ominus\star.\langle x^\ominus \parallel V\cdot\star\rangle \parallel e_\ominus\rangle \\
\pi_i\cdot e_\ominus &\stackrel{\text{def}^\ddagger}{=} \tilde{\mu}x^\ominus.\langle \mu^\ominus\star.\langle x^\ominus \parallel \pi_i\cdot\star\rangle \parallel e_\ominus\rangle
\end{aligned}
$$

(†: When t_+ is not a value—‡: When $e_\ominus$ is not a stack.)

Let us call *depolarisation* the following equation:

$$
\langle t_+ \parallel \tilde{\mu}x^+.c\rangle \rhd_\beta c[t_+/^f x^+]\,, \qquad (\beta)
$$

which is equivalent to the associativity of the composition in $\mathbf{L}_{\text{int}}$ (see [50]).

In $\mathbf{L}_{\text{int}}$ one can already derive modulo $\simeq_{\text{RE}}$ all the conversions of $\Lambda^{\vdash}$ with the following restriction: the arguments of λ and δ are restricted to values. Using (β) one can substitute positive variables, which are values, with arbitrary positive expressions. Therefore, $\Lambda^{\vdash}$ and $\mathbf{L}^{\vdash}_{\text{int}}$ are isomorphic modulo depolarisation:

Proposition 3. *If* $t \approx u$ *in* $\Lambda^{\vdash}$ *then* $\mathrm{S}[\![t]\!] \simeq_{\text{RE}\beta} \mathrm{S}[\![u]\!]$ *in* $\boldsymbol{L}^{\vdash}_{int}$.

In fact in (untyped) $\mathbf{L}_{\text{int}}$ extended with (β) one can apply the same diagonal argument as in Section III-A2, and therefore we know that (β) is incoherent in $\mathbf{L}_{\text{int}}$, as we claimed. But, (β) is compatible with $\mathbf{L}^{\vdash}_{\text{int}}$ at least in the following sense:

Proposition 4. $\boldsymbol{L}^{\vdash}_{int}$ *satisfies* (β) *modulo* $\simeq_{\text{RE}}$ *whenever* t_+ *is closed.*

This is an immediate consequence of the following:

Lemma 5. *Let* t_+ *be an expression such that* $\vdash_{\boldsymbol{L}_{int}} t_+ : B_+$. *There exists a closed value* V_+ *such that* $\langle t_+ \parallel \star\rangle \rhd^*_{\text{R}} \langle V_+ \parallel \star\rangle$, *and therefore* $t_+ \simeq_{\text{RE}} V_+$.

The existence of V_+ follows from polarised realizability, for instance (an immediate adaptation of) [47]. The second part is due to the linearity restriction on $\star$, which implies $\star \notin \mathbf{fv}(t_+, V_+)$.

Stronger forms of Proposition 4, extended to open terms modulo observational equivalence, to more expressive logics, and to other positive connectives, is the subject of further work. There is an analogy between this approach and the dinatural approach [29] to "*free theorems*": both rely on the fine knowledge of cut-elimination in sequent calculus, and on its termination, to work around the absence of a Curry-style λ-calculus with extensional sums.

Here the novelty of polarisation is to offer a Curry-style approach to typed λ-calculi where η-rules are valid *a priori*. For this we take the point of view that the associativity of composition in typed λ-calculi is an external property similar to normalisation (rather than the η-rules themselves being obtained a posteriori). From this point of view we cannot be surprised that the rewriting theory of the λ-calculus with extensional sums is complex, and that it can even be computationally inconsistent in the presence of fixed-points.

IV. The Structure of Normal Forms

A. The phase structure of normal forms

Figure 7a describes the syntax of *valid* $\rhd_{\text{R}}$-normal forms, where a constructor is never opposed to an abstractor for another connective. For example $\langle \iota_1(t) \parallel \tilde{\mu}(x,y).c\rangle$ would be an invalid normal form. A normal command of the form $\langle V_+ \parallel \star\rangle$ or $\langle x^\ominus \parallel S_\ominus\rangle$ starts with a *constructor*: head reduction is finished.

$$\langle V_+ \parallel \star \rangle$$
$$\langle x^+ \parallel e_+ \setminus \tilde{\mu} x^+.c \rangle$$
$$\langle x^\ominus \parallel S_\ominus \rangle$$
$$\langle t_\ominus \setminus \mu^\ominus \star.c \parallel \star \rangle$$

(a) Valid $\triangleright_R$-normal forms

$$f = c^\phi ::= \langle x \parallel S^\phi \rangle \mid \langle V^\phi \parallel \star \rangle \quad \text{where} \quad \phi ::= C \mid A \ \text{(phase: constructors, abstractors)}$$

$$V^\phi \begin{cases} V^C & ::= x^+ \mid (V^{C\pm}, V^{C\pm}) \mid \iota_1(V^{C\pm}) \mid \iota_2(V^{C\pm}) \\ V^A & ::= x^\ominus \mid \mu(x \cdot \star).f \mid \mu{<}\star.f\,;\star.f{>} \end{cases} \qquad S^\phi \begin{cases} S^A & ::= \star \mid \tilde{\mu}[x.f \mid y.f] \mid \tilde{\mu}(x,y).f \\ S^C & ::= \star \mid V^{C\pm}.S^{C\pm} \mid \pi_i \cdot S^{C\pm} \end{cases}$$

$$V^{C\pm} ::= V^C \mid \mu^\ominus \star.f \qquad S^{C\pm} ::= S^C \mid \tilde{\mu} x^+.f$$

(b) Focused forms

Figure 7 – Normal and focused forms

A command of the form $\langle x^+ \parallel e_+ \setminus \tilde{\mu} x^+.c \rangle$ (any e_+ that is not a $\tilde{\mu} x^+.c$) or $\langle t_\ominus \setminus \mu^\ominus \star.c \parallel \star \rangle$ starts with an *abstractor* whose computation is blocked by the lack of information on the opposite side. We therefore decompose the structure of normal commands as a succession of phases, with *constructor phases* c^C and *abstractor phases* c^A.

Figure 7b describes the structure of *phase-separated* $\to_R$-normal forms. We call them *focused* forms. A focused form f is a *focused command* c^ϕ for some *phase variable* ϕ, which is either C or A—we will write $\bar{\phi}$ for the phase opposite to ϕ. A focused command c^ϕ is either a *focused value* V^ϕ against $\star$, or a *focused stack* S^ϕ against some variable x. Constructor values V^C start with a head expression constructor, followed by other head constructors until $\mu^\ominus \star.f$ or a variable is reached, ending the phase. Constructor stacks S^C are non-empty compositions of stack constructors ended by a $\tilde{\mu} x^+.f$. Abstractor values or stacks are either a variable or $\star$, or a μ-form that immediately ends the phase.

This syntactic phase separation resembles *focused proofs* [4]. We work in an untyped calculus, so we cannot enforce that phases be maximally long; but otherwise constructor phases correspond to *non-invertible*, or *synchronous* rules, while abstractor phases correspond to *invertible*, or *asynchronous* inference rules.

Not all $\to_R$-normal forms are focused forms in the sense of Figure 7b because they may not respect the constraint that *phase change* must be explicitly marked by a μ or $\tilde{\mu}$ (for example with $\langle \iota_1(x^\ominus) \parallel \star \rangle$). But a normal form can be easily rewritten into a focused form by trivial $(\to_{E\mu\tilde{\mu}})$-expansions ($\left\langle \iota_1(\mu^\ominus \star.\langle x^\ominus \parallel \star \rangle) \,\middle\|\, \star \right\rangle$ in our example).

Lemma 6. *Any valid* $\to_R$*-normal form* $(\to^*_{E_{\mu,\tilde{\mu}}})$*-expands to a unique focused form* f.

B. Algorithmic equality for expansion rules

a) Restriction to SN terms: We study the equivalence of $\to_R$-normal forms modulo $\to_R$-strongly-normalising (SN) conversions. Therefore we set ourselves in the subset $\mathbf{L}^{SN}_{int}$ of SN terms of $\mathbf{L}_{int}$. This is a more general alternative to studying the equivalence modulo well-typed conversions for some type system. Notice that since $\to_R$ is confluent, any SN term has a unique $\to_R$-normal form.

$\mathbf{L}^{SN}_{int}$ is not closed under $\simeq_{RE}$, since any strongly-normalizable command is equivalent to some non-normalizable commands. We define $\triangleright^{SN}_R$ and $\triangleright^{SN}_E$ by restriction of $\triangleright_R$ and $\triangleright_E$ to $\mathbf{L}^{SN}_{int}$. The SN-compatible closure $\to^{SN}$ of $\triangleright^{SN}$ is defined such that $p \to^{SN} p'$ for p, p' two SN terms whenever p' is obtained from p by application of $\triangleright^{SN}$ on one of its sub-commands. Then $\simeq^{SN}$ is defined as the equivalence closure of $\to^{SN}$.

b) Algorithmic equality: Now we show that $\simeq^{SN}_{RE}$ is decidable, by defining a form of *algorithmic equality* on normal forms, and we deduce that $\simeq^{SN}_{RE}$ is not inconsistent.

We define the algorithmic equality relation $\equiv_A$ as a system of inference rules on pairs of SN commands, which captures extensional equivalence: $\equiv_A$ decides $\simeq^{SN}_E$. If we had worked on typed normal forms, we could have performed some maximal, type-directed, $\to_E$-expansion. However, for the sake of generality and its intrinsic interest, we work with Curry-style normal forms.

The purely syntactic approach performs, in both of the commands to be compared, the $\to_E$-expansions for the abstractors that appear in either command. This untyped equivalence is inspired by previous works (Coquand [10]; Abel and Coquand [1]) where the η-equivalence of λ-terms is computed by looking at the terms only, by doing an η-expansion when either term starts with λ.

In Figure 8a, we define $\equiv_A$ on SN commands by mutual recursion with a sub-relation $\equiv_{A_N}$, the latter of which only relates $\to_R$-normal forms.

We have omitted the symmetric counterparts for the abstractor rules. In the last two cases, we consider that $\equiv_A$ is defined on expressions and contexts as the congruence closure of $\equiv_A$ on sub-commands. For example, $\iota_1(\mu\star.c) \equiv_A t'$ if and only if t' is of the form $\iota_1(\mu\star.c')$ for some c' such that $c \equiv_A c'$. We use the term *algorithmic* because this equivalence is *almost* syntax-directed. If both sides of the equivalence start with an abstractor application, there is a non-syntax-directed choice of which side to inspect first: for t, u fixed there may be several distinct derivations of $t \equiv_A u$.

Algorithmic equivalence is well-defined because since the commands in the conclusion of the form $\equiv_{A_N}$ are normal, the commands in their premises are SN, hence they can be compared through $\equiv_A$. Indeed, whenever c is a $\to_R$-normal form, the substitutions applied by those rules (for instance $c[(x_1, x_2)/x^+]$) may create new redexes (for instance with $c = \langle x^+ \parallel \mu(y_1, y_2).c' \rangle$), but contracting such redexes only creates substitutions of variables for variables ($[x_1/y_1]$ and $[x_2/y_2]$ in this example). In particular any reduction path has a length of at most the number of occurrences of the variable (x^+ in the example).

Lemma 7. *Soundness of algorithmic equivalence: if* $t \equiv_A u$ *then* $t \simeq^{SN}_{RE} u$.

C. Completeness of algorithmic equality on normal forms

Abel and Coquand [1] discuss the problem of transitivity of their syntax-directed algorithmic equality: while it morally captures our intuition of valid η-equivalences, it fails to account for some seemingly-absurd equivalences that can be derived on untyped terms, such as $\lambda x.(t\,x) \simeq_\eta {<}\pi_1 t; \pi_2 t{>}$, the transitive

$$\frac{c \to_{\mathrm{R}}^{*} c_1(\twoheadrightarrow) \qquad c_1 \equiv_{\mathrm{A}_N} c_1' \qquad (\twoheadleftarrow)c_1' \leftarrow_{\mathrm{R}}^{*} c'}{c \equiv_{\mathrm{A}} c'}\ \text{A-R}$$

$$\frac{c[(x_1,x_2)/x^+] \equiv_{\mathrm{A}} c'[(x_1,x_2)/x^+]}{\langle x^+ \parallel \mu(x_1,x_2).c\rangle \equiv_{\mathrm{A}_N} c'}\ \text{A-pair} \qquad \frac{c_1[\iota_1(x_1)/x^+] \equiv_{\mathrm{A}} c'[\iota_1(x_1)/x^+] \qquad c_2[\iota_2(x_2)/x^+] \equiv_{\mathrm{A}} c'[\iota_2(x_2)/x^+]}{\langle x^+ \parallel \tilde{\mu}[x_1.c_1 | x_2.c_2]\rangle \equiv_{\mathrm{A}_N} c'}\ \text{A-sum}$$

$$\frac{c[(x\cdot\star)/\alpha] \equiv_{\mathrm{A}} c'[(x\cdot\star)/\star]}{\langle \mu(x\cdot\star).c \parallel \alpha\rangle \equiv_{\mathrm{A}_N} c'}\ \text{A-fun} \qquad \frac{c_1[\pi_1\cdot\star/\star] \equiv_{\mathrm{A}} c'[\pi_1\cdot\star/\star] \qquad c_2[\pi_2\cdot\star/\star] \equiv_{\mathrm{A}} c'[\pi_2\cdot\star/\star]}{\langle \mu{<}\star.c_1 ; \star.c_2{>} \parallel \star\rangle \equiv_{\mathrm{A}_N} c'}\ \text{A-neg-pair}$$

$$\frac{\pi_1^C \equiv_{\mathrm{A}} \pi_2^C}{\langle x^{\ominus} \parallel \pi_1^C\rangle \equiv_{\mathrm{A}_N} \langle x^{\ominus} \parallel \pi_2^C\rangle}\ \text{A-cocong} \qquad \frac{V_1^C \equiv_{\mathrm{A}} V_2^C}{\langle V_1^C \parallel \star\rangle \equiv_{\mathrm{A}_N} \langle V_2^C \parallel \star\rangle}\ \text{A-cong}$$

(a) Algorithmic equivalence

$$\frac{c[(x_1,x_2)/V_+] \equiv_{\mathrm{A}+} c'[(x_1,x_2)/V_+]}{\langle V_+ \parallel \mu(x_1,x_2).c\rangle \equiv_{\mathrm{A}+_N} c'}\ \text{A+pair} \qquad \frac{c_1[\iota_1(x_1)/V_+] \equiv_{\mathrm{A}} c'[\iota_1(x_1)/V_+] \qquad c_2[\iota_2(x_1)/V_+] \equiv_{\mathrm{A}} c'[\iota_2(x_1)/V_+]}{\langle V_+ \parallel \tilde{\mu}[x_1.c_1 | x_2.c_2]\rangle \equiv_{\mathrm{A}+_N} c'}\ \text{A+sum}$$

$$\frac{c[(x\cdot\star)/S_{\ominus}] \equiv_{\mathrm{A}+} c'[(x\cdot\star)/S_{\ominus}]}{\langle \mu(x\cdot\star).c \parallel S_{\ominus}\rangle \equiv_{\mathrm{A}+_N} c'}\ \text{A+fun} \qquad \frac{c_1[\pi_1\cdot\star/S_{\ominus}] \equiv_{\mathrm{A}+} c'[\pi_1\cdot\star/S_{\ominus}] \qquad c_1[\pi_2\cdot\star/S_{\ominus}] \equiv_{\mathrm{A}+} c'[\pi_2\cdot\star/S_{\ominus}]}{\langle \mu{<}\star.c_1 ; \star.c_2{>} \parallel S_{\ominus}\rangle \equiv_{\mathrm{A}+_N} c'}\ \text{A+negpair}$$

(b) Extended algorithm equality rules

$$\frac{\dfrac{\dfrac{\langle \iota_1(y_1) \parallel \star\rangle \equiv_{\mathrm{A}+_N} \langle \iota_1(y_1) \parallel \star\rangle \qquad \langle \iota_2(y_1) \parallel \star\rangle \equiv_{\mathrm{A}+_N} \langle \iota_2(y_1) \parallel \star\rangle}{\langle (x_1,x_2) \parallel \star\rangle \equiv_{\mathrm{A}+_N} \left\langle (x_1,x_2) \,\middle\|\, \tilde{\mu}\left[y_1.\langle (x_1,x_2) \parallel \star\rangle \middle| y_2.\langle (x_1,x_2) \parallel \star\rangle\right]\right\rangle}\ \text{A+sum}}{\left\langle x^+ \,\middle\|\, \mu(x_1,x_2).\langle x^+ \parallel \star\rangle\right\rangle \equiv_{\mathrm{A}+_N} \left\langle x^+ \,\middle\|\, \tilde{\mu}\left[y_1.\langle x^+ \parallel \star\rangle \middle| y_2.\langle x^+ \parallel \star\rangle\right]\right\rangle}\ \text{A+pair}}{(\mathsf{let}\,(x_1,x_2)\,\mathsf{be}\,x^+\,\mathsf{in}\,x^+) \equiv_{\mathrm{A}+_N} \delta(x^+,\iota_1(y_1).x^+,\iota_2(y_2).x^+)}$$

$$\frac{\dfrac{\dfrac{\langle (x_1,x_2) \parallel y\cdot\star\rangle \equiv_{\mathrm{A}+_N} \langle (x_1,x_2) \parallel y\cdot\star\rangle}{\left\langle x^+ \,\middle\|\, \mu(x_1,x_2).\langle (x_1,x_2) \parallel y\cdot\star\rangle\right\rangle \equiv_{\mathrm{A}+_N} \langle x^+ \parallel y\cdot\star\rangle}\ \text{A+pair}}{\left\langle x^+ \,\middle\|\, \mu(x_1,x_2).\langle (x_1,x_2) \parallel \star\rangle\right\rangle \equiv_{\mathrm{A}+_N} \left\langle \mu(y\cdot\star).\langle x^+ \parallel y\cdot\star\rangle \,\middle\|\, \star\right\rangle}\ \text{A+fun}}{(\mathsf{let}\,(x_1,x_2)\,\mathsf{be}\,x^+\,\mathsf{in}\,(x_1,x_2)) \equiv_{\mathrm{A}+_N} \lambda y.(x^+ y)}$$

(c) Deriving (instances of) Abel and Coquand's [1] transitivity rules from $\equiv_{\mathrm{A}+}$.

Figure 8 – Algorithmic equivalence

combination of $\lambda x.(t\,x) \simeq_\eta t$ and $t \simeq_\eta <\pi_1 t ; \pi_2 t>$. To account for those absurd equivalences, Abel and Coquand had to extend their relation with type-incorrect rules such as:

$$\frac{t \equiv_{\mathrm{A}} n\,x \qquad \pi_1 n \equiv_{\mathrm{A}} r \qquad \pi_2 r \equiv_{\mathrm{A}} s}{\lambda x.(t\,x) \equiv_{\mathrm{A}} <r ; s>}$$

This change suffices to regain transitivity and capture untyped η-conversion. Unfortunately, this technique requires a number of additional rules quadratic in the number of different constructors of the language. Our richer syntax allows the definition in Figure 8b of an *extended algorithmic equivalence* $\equiv_{\mathrm{A}+}$ without introducing any new rule. Instead, we extend each constructor rule A-fun, A-pair, etc., in a simple (but potentially type-incorrect) way.

We introduce the notation $c[t/V]$ which substitutes syntactic occurrences of the term V by t—there is a unique decomposition of c into $c'[V/x]$ with $V \notin c'$, and then $c[t/V]$ is defined as $c'[t/x]$. On valid terms, this extended equivalence coincides with the previous algorithmic equivalence. Indeed, in the case of a normal command $\langle V_+ \parallel \mu(x_1,x_2).c\rangle$ for example, a valid V_+ can only be a variable x^+, in which case the rule A-pair and A+pair are identical. The extension only applies when V_+ has a head constructor that is not the pair, in which case the expression is in normal form but invalid.

Figure 8c shows that this extension subsumes the rules of Abel and Coquand. Notice how the inference following the leftmost leaf is obtained by the apparently-nonsensical substitution $\langle (x_1,x_2) \parallel \star\rangle [\iota_1 y_1 / (x_1,x_2)]$ produced by the extended sum rule on an invalid command.

Lemma 8 (Soundness)**.** *One has* $\equiv_{\mathrm{A}+} \subseteq \simeq_{\mathrm{RE}}^{\mathrm{SN}}$.

Lemma 9 (Reflexivity)**.** *If t is SN then $t \equiv_{\mathrm{A}+} t$.*

Lemma 10 (Transitivity)**.** *If $t_1 \equiv_{\mathrm{A}+} t_2$ and $t_2 \equiv_{\mathrm{A}+} t_3$ then $t_1 \equiv_{\mathrm{A}+} t_3$.*

Lemma 11 (Completeness)**.** *If $c \simeq_{\mathrm{RE}}^{\mathrm{SN}} c'$ then $c \equiv_{\mathrm{A}+} c'$.*

Theorem 12. *$c_1 \simeq_{\mathrm{RE}}^{\mathrm{SN}} c_2$ if and only if $c_1 \equiv_{\mathrm{A}+} c_2$.*

Corollary 13. *$\simeq_{\mathrm{RE}}^{\mathrm{SN}}$ is consistent.*

Indeed, $\langle x^+ \parallel \star\rangle$ is not algorithmically equivalent to $\langle y^+ \parallel \star\rangle$. This result on $\simeq_{\mathrm{RE}}^{\mathrm{SN}}$ does not allow us to prove our conjecture that $\simeq_{\mathrm{RE}}$ is consistent: notice that in the case of the untyped λ-calculus with sums, the inconsistency indeed relies on non-normalizing intermediate conversion steps, involving the fixed point of $\lambda x.\delta(x, y.\iota_2(y), y.\iota_1(y))$ as we recalled in the introduction.

$$
\begin{array}{rl}
 & c^\phi[\,] ::= \langle x \parallel S^\phi[\,]\rangle \mid \langle V^\phi[\,] \parallel \star\rangle \\
V^\phi & \begin{cases} V^C[\,] ::= (V^{C\pm}, V^{C\pm}[\,]) \mid (V^{C\pm}[\,], V^{C\pm}) \mid \iota_i(V^{C\pm}[\,]) \\ V^A[\,] ::= \mu(x{\cdot}\star).[\,] \mid \mu{<}\star.[\,]; \star.[\,]{>} \end{cases} \\
 & V^{C\pm} ::= V^C[\,] \mid \mu^{\ominus}\star.[\,] \\
S^\phi & \begin{cases} S^A ::= \tilde{\mu}[x.[\,]|x.[\,]] \mid \tilde{\mu}(x,y).[\,] \\ S^C ::= V^{C\pm}[\,]{\cdot}S^{C\pm} \mid V^{C\pm}{\cdot}S^{C\pm}[\,] \mid \pi_i{\cdot}S^{C\pm}[\,] \\ S^{C\pm} ::= S^C[\,] \mid \tilde{\mu}x^+.[\,] \end{cases}
\end{array}
$$

Figure 9 – Commutative contexts

D. Phase equivalence

As we have seen, $\simeq^{\mathrm{SN}}_{\mathrm{RE}}$ is more restrictive than the usual $\beta\eta$-equivalence considered for λ-terms with sums. For example, $\mathsf{let}\, y^+ \,\mathsf{be}\, x\, V \,\mathsf{in}\, \iota_1(t_\ominus)$ is not $\simeq^{\mathrm{SN}}_{\mathrm{RE}}$-equivalent to $\iota_1(\mathsf{let}\, y^+ \,\mathsf{be}\, x\, V \,\mathsf{in}\, t_\ominus)$. Since the goal is to decide $\beta\eta$ in the λ-calculus with sums, we consider the equation (β) of depolarisation on $\mathbf{L}^{\mathrm{SN}}_{\mathrm{int}}$. On focused forms, this allows identical phases to be merged, and unused phases to be dropped. First, we study in this section the reordering of independent phases. This captures the global aspect of equivalence. After this we will be able to define merging and weakening locally.

We write $c^\phi[\ ]$, $V^\phi[\ ]$, and $S^\phi[\ ]$ for *focused command contexts*, *value contexts*, and *stack contexts* with one or several variable-capturing holes standing for subcommands. We are interested in reordering $c_1[c_2[\,]]$ into $c_2[c_1[\,]]$. Two command contexts $c_1[\ ]$ and $c_2[\ ]$ are *independent*, written $c_1 \# c_2$, if reordering them respects the scope: the variables (and $\star$) of c_1 bound by c_1 are not free in c_2, and conversely.

Definition 14. We define the *commutative contexts* for focused commands, values and stacks by the grammar of Figure 9. For a given phase ϕ, we define the phase equivalence $\simeq_{\mathrm{P}_\phi}$ with the following reordering whenever $c_1^\phi \# c_2$:

$$c_2[c_1^\phi[\,]] \rhd_{\mathrm{P}_\phi} c_1^\phi[c_2[\,]]$$

Note that some abstractor commutative contexts, for instance $\langle x^+ \parallel \tilde{\mu}[x_1.[\,]|x_2.[\,]]\rangle$, may have several holes—phase reordering may therefore result in (de-)duplication.

Reordering independent abstractor commutative contexts is always sound with respect to $\simeq^{\mathrm{SN}}_{\mathrm{RE}}$. On the contrary, reordering independent constructor commutative contexts requires the depolarized equivalence $\simeq^{\mathrm{SN}}_{\mathrm{RE}\beta}$.

Lemma 15. *One has* $\simeq_{\mathrm{P}_A} \subseteq \simeq^{\mathrm{SN}}_{\mathrm{RE}}$.

Lemma 16. *One has* $\simeq_{\mathrm{P}_C} \subseteq \simeq^{\mathrm{SN}}_{\mathrm{RE}\beta}$.

Although $\simeq_{\mathrm{P}_\phi}$ is defined entirely on the structure of focused normal forms, the intermediary $\simeq^{\mathrm{SN}}_{\mathrm{RE}}$-reasoning step we use to justify commutativity are free to ignore the focusing constraints. These exchanges between the normalized and the non-normalized world are interesting and justify, we think, studying first the dynamics of the system (proof/type systems with an explicit cut), and looking at their normal forms only later, instead of starting with cut-free systems directly.

Definition 17. We call *phase equivalence* the equivalence relation $\simeq_{\mathrm{P}}$ obtained from the union of $\simeq_{\mathrm{P}_C}$ and $\simeq_{\mathrm{P}_A}$.

The phase equivalence can be decided by splitting it into a *local* equivalence relation $\simeq_{\mathrm{LP}}$ that only permutes context of the same phase, and a *global* (ordered) rewriting relation $\rightarrow_{\mathrm{GP}}$ that permutes context across maximal contexts of the opposite phase—this is inspired from the *preemptive rewriting* relation of maximal multi-focusing [9, 57]. (See Appendix A.)

E. Recovering $\beta\eta$-equivalence

We now go strictly beyond the equivalences induced by multi-focusing presented in the previous sections, and introduce *merging* and *weakening* simplification to recover full $\beta\eta$-equivalence.

If ρ is a substitution from variables to variables, and $V^\phi[\ ]$ (resp. $S^\phi[\ ]$) is a focused context, we write $V^\phi[\rho[\,]]$ for the focused context where each of the variables bound by $V^\phi[\ ]$ that scope over its hole are renamed according to ρ, and only those are in the domain of ρ. In particular, there exists a ρ such that $V_1^\phi[\rho[\,]] = V_2^\phi[\,]$ when V_1^ϕ and V_2^ϕ modulo renaming of outgoing variables—for example $\mu(x_1, x_2).[\,]$ and $\mu(x_1', x_2').[\,]$.

Definition 18. We define the *merging simplification* $\rhd_{\mathrm{M}}$ such that, for each pair of *independent* commutative contexts $c_1^\phi \# c_2^\phi$ and substitution ρ such that $c_1^\phi = c_2^\phi[\rho]$, we have $c_1^\phi[c_2^\phi[c_3]] \rhd_{\mathrm{M}} c_1^\phi[\rho[c_3]]$.

Definition 19. We define the *weakening simplification* $\rhd_{\mathrm{W}}$ such that, whenever none of the variables bound by the commutative context $c[\,]$ over its hole are used in f, we have $c[f] \rhd_{\mathrm{W}} f$.

Note that commutative contexts of the form $\langle V^\phi[\,] \parallel \star\rangle$ are neither mergeable nor weakenable, since all possible holes of $V^\phi[\ ]$ shadow the variable $\star$, breaking the independence requirement, and all pluggable commands f use the variable $\star$, breaking the weakening requirement. Semantically it would make no sense, of course, to claim something to the effect of $\iota_1(\iota_1(\mu\star.c)) \simeq_{\mathrm{M}} \iota_1(\mu\star.c)$ or $\iota_1(\mu\star.f) \simeq_{\mathrm{W}} f$, but we directly recover this from $\mathbf{L}_{\mathrm{int}}$ without having to break the uniformity of our definitions.

We will write $\simeq_{\mathrm{APMW}}$ the closure of the unions of $\equiv_{\mathrm{A}+}$, $\simeq_{\mathrm{P}}$, $\simeq_{\mathrm{M}}$ and $\simeq_{\mathrm{W}}$. It is decidable on SN terms by normalizing, then taking the $\simeq_{\mathrm{P}}$-normal form, the $\rightarrow_{\mathrm{MW}}$-normal form, and finally $\equiv_{\mathrm{A}+}$ in order to decide the $\simeq^{\mathrm{SN}}_{\mathrm{RE}}$-equivalence. We now prove that it is sound and complete with respect to $\simeq^{\mathrm{SN}}_{\mathrm{RE}\beta}$, which essentially corresponds to the $\beta\eta$-equality of simply-typed λ-calculus as demonstrated in Section III-C.

Proposition 20. *One has* $\simeq_{\mathrm{MW}} \subseteq \simeq^{\mathrm{SN}}_{\mathrm{RE}\beta}$.

Theorem 21. *One has* $\simeq^{\mathrm{SN}}_{\mathrm{RE}\beta} \subseteq \simeq_{\mathrm{APMW}}$.

This concludes our study of the normal forms. We have demonstrated that the equivalence $\simeq^{\mathrm{SN}}_{\mathrm{RE}\beta}$, which corresponds on typed terms to the natural equality of well-typed λ-terms, can be re-defined, and decided, in a syntactically uniform way based on sequences of constructor and abstractor phases.

V. Discussion

A. Liang and Miller's LJF

This section complements the relationship with focusing established in Section IV-A by restricting to simply-typed $\mathbf{L}_{\mathrm{int}}$.

Focused systems describe (various notions of) η-long, cut-free proofs. We sketch how Liang and Miller's focused system LJF [43, 44] characterises $\rightarrow_{\mathrm{E}}$-long, $\rightarrow_{\mathrm{R}}$-normal derivations of simply-typed $\mathbf{L}_{\mathrm{int}}$. We omit first-order quantifiers and units, which can easily be added in simply-typed $\mathbf{L}_{\mathrm{int}}$.

Any term in simply-typed $\mathbf{L}_{\text{int}}$ is equivalent via $\to_{\text{E}}$ to a term where all the variables x^P and co-variables $\star^N$, for N and P non-atomic, are decomposed using transformations of sub-commands c of the following form:

$$P = A \otimes B: \quad \langle x^{A\otimes B} \parallel \tilde{\mu}(y,z).c[(y^A, z^B)/x]\rangle$$
$$P = A \oplus B: \quad \left\langle x^{A\oplus B} \,\middle\|\, \tilde{\mu}\left[y.c[\iota_2(y^A)/x]\middle|z.c[\iota_2(z^B)/x]\right]\right\rangle$$
$$N = A \to B: \quad \langle \mu(x\cdot\star).c[x^A\cdot\star^B/\star] \parallel \star^{A\to B}\rangle$$
$$N = A \times B: \quad \langle \mu{<}\star.c[\pi_1\cdot\star^A/\star]; \star.c[\pi_2\cdot\star^B/\star]{>} \parallel \star^{A\times B}\rangle$$

Also, variables x^N and co-variables $\star^P$ are treated respectively like $\mu\star^N.\langle x \parallel \star\rangle$ and $\tilde{\mu}x^P.\langle x \parallel \star\rangle$ in this respect, like in Section IV-A.

Any $\to_{\text{R}}$-normal term in simply-typed $\mathbf{L}_{\text{int}}$ is equivalent to a $\to_{\text{R}}$-normal term where the variables are decomposed as above. Indeed, as we already observed in Section IV-B, if c above is normal, then the length of any normalisation is bounded by the number of occurrences of x and $\star$. We define a $\to_{\text{R}}$-normal term to be $\to_{\text{E}}$-long when the variables are decomposed in this way as early as possible, in any order. Note that $\to_{\text{E}}$-long, $\to_{\text{R}}$-normal term are not necessarily canonical.

Judgements of simply-typed $\mathbf{L}_{\text{int}}$ correspond to the ones of LJF as per the following table, where terms are $\to_{\text{R}}$-normal and $\to_{\text{E}}$-long, where formulae in Γ are negative or atomic, and where D is positive or atomic:

LJF	Simply-typed $\mathbf{L}_{\text{int}}$
$[\Gamma], \Theta \to \mathcal{R}$	$c : (\Gamma, \Theta \vdash \star : R)$
$[\Gamma] \xrightarrow[A]{} [D]$	$\Gamma \mid S : A \vdash \star : D$
$[\Gamma] \xrightarrow[A]{}$	$\Gamma \vdash V : A$

Making the above claim formal can lead to a proof of completeness of LJF, the closest instance of which would currently be Simmons' [59].

Liang and Miller suggest that one can "*examine the impact of* [LJF] *on typed λ-calculi*", *"the LJF system could provide a framework for λ-term evaluations that combine the eager and lazy evaluation strategies*" [44]. From this point of view, $\mathbf{L}_{\text{int}}$ can be seen as Curry-style LJF, where cut-elimination is made explicit as a process, and where expansions can be delayed until useful (for instance, as in our decision algorithm). For reasoning about programs and proofs, this makes a crucial difference: indeed, LJF, as it stands, assumes that one always want to apply the reduction ($R\mu$) and the expansion ($E\oplus$). This means that derivations are duplicated inside branches whenever there is an opportunity to do so.

In terms of intermediate representations, this recalls the difficulty of not duplicating code with commuting conversions, a reason among which Kennedy advocates CPS over monadic meta-languages and ANF [34]. $\mathbf{L}_{\text{int}}$ is closer to CPS, as we will see, and does not suffer from this issue.

B. Interpretations of focusing and polarisation

The notation with abstractors for invertible rules, for instance $\tilde{\mu}(x,y).c$, is well suited to suggest the corresponding expansion:

$$e_+ \rhd_{\text{R}} \tilde{\mu}(x,y).\langle (x,y) \parallel e_+\rangle$$

as well as the fact that abstractors compose:

$$\tilde{\mu}(x,(y,z)).c \stackrel{\text{def}}{=} \tilde{\mu}(x,a).\langle a \parallel \tilde{\mu}(y,z).c\rangle$$

An analogy between pattern-matching and asynchronous phases in focusing was introduced by Zeilberger [69, 70].

It appears that the analogy simply extends to calculi where focalisation is expressed as constraints on the reduction rather than on the derivation rules. It also extends to connectives whose focusing properties do not match the polarities, such as the exponentials of linear logic [47]. The pattern-matching approach to focalisation has also been further explored in the context of abstract-machine-like calculi in Curien and M.-M. [12].

See [50] for a further comparison of our notion of polarisation with Zeilberger's (and Laurent's [39]), in particular regarding the structure of shifts ($\Downarrow \dashv \Uparrow$ *vs.* $\uparrow \dashv \downarrow$).

C. Defunctionalised CPS in direct style

A continuation is the functional abstraction of the rest of a program [62, 55]. By this definition, there is little difference between the $\tilde{\mu}x.c$ construct and a continuation. In this sense, the idea of solving transitions of abstract machines subsumes the concept of continuation. In this section we replace abstract-machine-like calculi in the continuity of continuation-passing-style semantics.

a) Inside-out contexts and defunctionalisation: The inverted, non-functional representation of contexts appeared in continuation-based semantics as early as Wand [68]. Non-functional representations of contexts are also apparent in Lafont, Reus and Streicher's continuation semantics [35]. Inside-out contexts have only been introduced for term calculi later, independently, by Herbelin in his study of the intuitionistic sequent calculus [31].

Danvy and Nielsen [15] recognised in Wand's technique an instance of Reynold's defunctionalisation of continuations [54]. Moreover, they state the "*isomorphism*" between expressions with a hole and these inside-out contexts.

With abstract-machine-like calculi, we now understand that inside-out contexts are more primitive. Our arguments in Section II-H crucially rely on the discovery of μ and $\tilde{\mu}$. The μ binder is necessary for obtaining expressions with a hole that correspond to destructors, by solving the adjoint reductions. The $\tilde{\mu}$ binder, on the other hand, is used to show that any expression with a hole can already be represented as a context.

b) CPS, direct style, and evaluation order: In [48, Chapter III], we showed (however in the context of *shift/reset* control operators) how call-by-value and call-by-name CPS translations decompose through a polarised abstract-machine-like calculus. The first step corresponds to solving the source language in the abstract-machine calculus. This implies that the source language is *macro-expressible*, in a sense close to Felleisen [20], in the abstract-machine-like calculus, like we have shown for the λ-calculus with sums in Section III with the additional depolarisation equation. In this sense, the abstract-machine-like calculus decomposes continuation-passing style into direct style.

Also, the call-by-name function type is defined as $N \to M$ and the call-by-value function type is defined as $\Downarrow(P \to Q)$: they are Levy's translations $U\underline{N'} \to \underline{M'}$ and $U(P' \to FQ')$ [41] (where $'$ denotes the rest of the translation), further decomposed as macro-expressions in terms of the negative type $A \to B$. In $\mathbf{L}_{\text{int}}$ we can already express the call-by-value function type as $(P \to Q) \otimes (X \to X)$ for X a distinguished atom, however this does not satisfy all the desired expansions. By adding connectives for shifts it becomes possible to define the untyped

call-by-value restriction of the λ-calculus with sums in $\mathbf{L}_{\text{int}}$, by adapting [48].

c) Defunctionalisation, CPS, and abstract machines: Streicher and Reus *"automatically"* derive abstract machines from continuation semantics—*"the transition rules correspond to the semantic equations"* [63]. Ager, Biernacki, Danvy, and Midtgaard [2] systematically transform evaluators into abstract machines, using CPS transformation, closure conversion, and defunctionalisation. Puech [53] reconstructs intuitionistic sequent calculus from natural deduction by adapting this technique.

Thus, from the perspective of continuations, the technique of abstract-machine-like calculi can be described as *defunctionalised CPS in direct style*. (Explicit closures have been left aside by immediately considering substitution.) However, our perspective takes the opposite direction, by assuming that abstract machines come first, and by not assuming the prior knowledge of CPS transformations.

Acknowledgements

The sections II and III are adapted from the first author's PhD thesis [48, Chapter I]. Many thanks to Pierre-Évariste Dagand, Olivier Danvy, Tim Griffin, Adrien Guatto, Thomas Streicher, Noam Zeilberger, and the anonymous reviewers, for their comments, criticisms and suggestions. We gratefully acknowledge partial support from the ERC project Ecsym and from the French National Research Agency (ANR-12-JS02-006-01, ANR-10-BLAN-0213, ANR-11-BS02-0010).

References

[1] Andreas Abel and Thierry Coquand, *Untyped Algorithmic Equality for Martin-Löf's Logical Framework with Surjective Pairs*, Proc. TLCA (Paweł Urzyczyn, ed.), LNCS, vol. 3461, Springer, 2005, pp. 23–38 (English).

[2] Mads Sig Ager, Dariusz Biernacki, Olivier Danvy, and Jan Midtgaard, *A Functional Correspondence between Evaluators and Abstract Machines*, Proc. PPDP, 2003, pp. 8–19.

[3] Thorsten Altenkirch, Peter Dybjer, Martin Hofmann, and Philip Scott, *Normalization by evaluation for typed lambda calculus with coproducts*, Proc. LICS, 2001, pp. 303–310.

[4] Jean-Marc Andreoli, *Logic Programming with Focusing Proof in Linear Logic*, Journal of Logic and Computation **2** (1992), no. 3, 297–347.

[5] Zena M Ariola, Aaron Bohannon, and Amr Sabry, *Sequent calculi and abstract machines*, ACM Trans. Program. Lang. Syst. **31** (2009), no. 4, 13.

[6] Vincent Balat, Roberto Di Cosmo, and Marcelo P. Fiore, *Extensional normalisation and type-directed partial evaluation for typed lambda calculus with sums*, Proc. POPL, 2004, pp. 64–76.

[7] Aloïs Brunel, *Quantitative classical realizability*, Inf. Comput. (2014), to appear.

[8] Kaustuv Chaudhuri, Stefan Hetzl, and Dale Miller, *A multi-focused proof system isomorphic to expansion proofs*, Journal of Logic and Computation (2014).

[9] Kaustuv Chaudhuri, Dale Miller, and Alexis Saurin, *Canonical sequent proofs via multi-focusing*, Proc. IFIP TCS, Springer, 2008, pp. 383–396.

[10] Thierry Coquand, *An Algorithm for Testing Conversion in Type Theory*, Logical Frameworks (Gérard Huet and Gordon Plotkin, eds.), Cambridge University Press, New York, NY, USA, 1991, pp. 255–279.

[11] Pierre-Louis Curien and Hugo Herbelin, *The duality of computation*, ACM SIGPLAN Notices **35** (2000), 233–243.

[12] Pierre-Louis Curien and Guillaume Munch-Maccagnoni, *The duality of computation under focus*, Proc. IFIP TCS, 2010, Extended version.

[13] Vincent Danos, Jean-Baptiste Joinet, and Harold Schellinx, *A New Deconstructive Logic: Linear Logic*, Journal of Symbolic Logic **62 (3)** (1997), 755–807.

[14] Olivier Danvy and Kevin Millikin, *A Rational Deconstruction of Landin's SECD Machine with the J Operator*, Logical Methods in Computer Science **4** (2008), 1–87.

[15] Olivier Danvy and Lasse R. Nielsen, *Defunctionalization at Work*, Research Report BRICS RS-01-23, Department of Computer Science, Aarhus University, 2001, Extended version of an article presented at PPDP 2001.

[16] Daniel J. Dougherty, *Some Lambda Calculi With Categorical Sums and Products*, Proc. RTA, 1993.

[17] Paul Downen and Zena M Ariola, *The duality of construction*, Proc. ESOP, Springer, 2014, pp. 249–269.

[18] José Espírito Santo, *Completing Herbelin's programme*, Typed Lambda Calculi and Applications, Springer, 2007, pp. 118–132.

[19] José Espírito Santo, Ralph Matthes, and Luís Pinto, *Continuation-Passing Style and Strong Normalisation for Intuitionistic Sequent Calculi*, Logical Methods in Computer Science **5** (2009), no. 2.

[20] Matthias Felleisen, *On the expressive power of programming languages*, Science of computer programming **17** (1991), no. 1, 35–75.

[21] Andrzej Filinski, *Declarative Continuations and Categorical Duality*, Master's thesis, DIKU, Computer Science Department, University of Copenhagen, Aug 1989, DIKU Rapport 89/11.

[22] Gerhard Gentzen, *Untersuchungen über das logische Schließen*, Mathematische Zeitschrift **39** (1935), 176–210,405–431.

[23] Neil Ghani, *$\beta\eta$-equality for coproducts*, Proc. TLCA, Springer, 1995, pp. 171–185.

[24] Jean-Yves Girard, *A new constructive logic: Classical logic*, Math. Struct. Comp. Sci. **1** (1991), no. 3, 255–296.

[25] ———, *On the Unity of Logic*, Ann. Pure Appl. Logic **59** (1993), no. 3, 201–217.

[26] ———, *Locus Solum: From the Rules of Logic to the Logic of Rules*, Math. Struct. Comp. Sci. **11** (2001), 301–506.

[27] ———, *Le Point Aveugle, Cours de logique, Tome II: Vers l'imperfection*, Visions des Sciences, Hermann, 2007, published subsequently in English [28].

[28] ———, *The Blind Spot: Lectures on Logic*, European Mathematical Society, 2011.

[29] Jean-Yves Girard, Andre Scedrov, and Philip J. Scott, *Normal Forms and Cut-Free Proofs as Natural Transformations*, Proc. Log. from Comp. Sci., Springer-Verlag, 1992, pp. 217–241.

[30] Healfdene Goguen, *Justifying Algorithms for $\beta\eta$-Conversion*, Proc. FoSSaCS (Vladimiro Sassone, ed.), LNCS, vol. 3441, Springer, 2005, pp. 410–424 (English).

[31] Hugo Herbelin, *Lambda-calculus structure isomorphic to sequent calculus structure*, Proc. CSL, 1994.

[32] Hugo Herbelin, *C'est maintenant qu'on calcule, au cœur de la dualité*, 2005, Habilitation thesis.

[33] Hagen Huwig and Axel Poigné, *A Note on Inconsistencies Caused by Fixpoints in a Cartesian Closed Category*, Theor. Comput. Sci. **73** (1990), no. 1, 101–112.

[34] Andrew Kennedy, *Compiling with continuations, continued*, ICFP, 2007, pp. 177–190.

[35] Yves Lafont, B. Reus, and Thomas Streicher, *Continuation Semantics or Expressing Implication by Negation*, Tech. report, University of Munich, 1993.

[36] J. Lambek and P. J. Scott, *Introduction to higher order categorical logic*, Cambridge University Press, New York, NY, USA, 1986.

[37] Peter J. Landin, *The mechanical evaluation of expressions*, The Computer Journal **6** (1964), no. 4, 308–320.

[38] ———, *A correspondence between* ALGOL *60 and Church's Lambda-notation*, Commun. ACM **8** (1965), no. 2, 89–101 and 158–165.

[39] Olivier Laurent, *Etude de la polarisation en logique*, Ph.D. thesis, Université Aix-Marseille II, 2002.

[40] F. William Lawvere, *Diagonal arguments and cartesian closed categories*, Lecture Notes in Mathematics **92** (1969), 134–145.

[41] Paul Blain Levy, *Call-by-Push-Value: A Subsuming Paradigm*, Proc. TLCA '99, 1999, pp. 228–242.

[42] ———, *Call-By-Push-Value: A Functional/Imperative Synthesis*, Semantic Structures in Computation, vol. 2, Springer, 2004.

[43] Chuck Liang and Dale Miller, *Focusing and Polarization in Intuitionistic Logic*, Proc. CSL, 2007, pp. 451–465.

[44] ———, *Focusing and polarization in linear, intuitionistic, and classical logics*, Theor. Comput. Sci. **410** (2009), no. 46, 4747–4768.

[45] Sam Lindley, *Extensional rewriting with sums*, Proc. TLCA, Springer, 2007, pp. 255–271.

[46] Paul-André Melliès and Nicolas Tabareau, *Resource modalities in tensor logic*, Ann. Pure Appl. Logic **161** (2010), no. 5, 632–653.

[47] Guillaume Munch-Maccagnoni, *Focalisation and Classical Realisability*, Proc. CSL, LNCS, Springer-Verlag, 2009.

[48] ———, *Syntax and Models of a non-Associative Composition of Programs and Proofs*, Ph.D. thesis, Univ. Paris Diderot, 2013.

[49] ———, *Formulae-as-Types for an Involutive Negation*, Proc. CSL-LICS, 2014.

[50] ———, *Models of a Non-Associative Composition*, Proc. FoSSaCS (A. Muscholl, ed.), LNCS, vol. 8412, Springer, 2014, pp. 397–412.

[51] Tobias Nipkow, *Higher-Order Critical Pairs*, Proc. LICS, 1991, pp. 342–349.

[52] Michel Parigot, *Lambda-Mu-Calculus: An Algorithmic Interpretation of Classical Natural Deduction*, Proc. LPAR, 1992, pp. 190–201.

[53] Matthias Puech, *Proofs, Upside Down*, Programming Languages and Systems (Chung-chieh Shan, ed.), LNCS, vol. 8301, Springer International Publishing, 2013, pp. 365–380 (English).
[54] John C. Reynolds, *Definitional Interpreters for Higher-Order Programming Languages*, Proc. ACM Nat. Conf., 1972, Reprinted in Higher-Order and Symbolic Computation 11(4):363-397, 1998, with a foreword [56], pp. 717–740.
[55] _____, *The Discoveries of Continuations*, Lisp and Symbolic Computation **6** (1993), no. 3-4, 233–248.
[56] _____, *Definitional Interpreters Revisited*, Higher-Order and Symbolic Computation **11** (1998), no. 4, 355–361.
[57] Gabriel Scherer, *Multi-focusing on extensional rewriting with sums*, Proc. TLCA, 2015.
[58] Peter Selinger, *Control Categories and Duality: On the Categorical Semantics of the Lambda-Mu Calculus*, Math. Struct. Comp. Sci. **11** (2001), no. 2, 207–260.
[59] Robert J. Simmons, *Structural Focalization*, ACM Trans. Comput. Log. **15** (2014), no. 3, 21:1–21:33.
[60] Arnaud Spiwak, *A dissection of L*, Submitted, 2014.
[61] Kristian Støvring, *Extending the Extensional Lambda Calculus with Surjective Pairing is Conservative*, Logical Methods in Computer Science **2** (2006), no. 2.
[62] Christopher Strachey and Christopher P. Wadsworth, *Continuations: A Mathematical Semantics for Handling Full Jumps*, Technical Monograph PRG-11, Oxford University Computing Laboratory, Programming Research Group, 1974, Reprinted in Higher-Order and Symbolic Computation 13(1/2):135–152, 2000, with a foreword [67].
[63] Thomas Streicher and Bernhard Reus, *Classical Logic, Continuation Semantics and Abstract Machines*, J. Funct. Program. **8** (1998), no. 6, 543–572.
[64] Hayo Thielecke, *Categorical Structure of Continuation Passing Style*, Ph.D. thesis, University of Edinburgh, 1997.
[65] Adriaan van Wijngaarden, *Recursive definition of syntax and semantics*, Formal Language Description Languages for Computer Programming, North Holland Publishing Company, 1966.
[66] Philip Wadler, *Call-by-value is dual to call-by-name*, SIGPLAN Not. **38** (2003), no. 9, 189–201.
[67] Christopher P. Wadsworth, *Continuations revisited*, Higher-Order and Symbolic Computation **13** (2000), no. 1/2, 131–133.
[68] Mitchell Wand, *Continuation-Based Program Transformation Strategies*, J. ACM **27** (1980), no. 1, 164–180.
[69] Noam Zeilberger, *On the unity of duality*, Ann. Pure and App. Logic **153:1** (2008).
[70] _____, *The logical basis of evaluation order*, Ph.D. thesis, Carnegie Mellon University, 2009.

$$\begin{array}{ccc}
\left\langle x \,\middle\|\, \pi^{\phi}\left[\langle V^{\phi}[\,] \parallel \star\rangle\right] \right\rangle & \simeq_{\mathrm{LP}} & \left\langle V^{\phi}\left[\langle x \parallel S^{\phi}[\,]\rangle\right] \,\middle\|\, \star \right\rangle \\
\left\langle x \,\middle\|\, S^{\phi}\left[\langle y \parallel S_2^{\phi}[\,]\rangle\right] \right\rangle & \simeq_{\mathrm{LP}} & \left\langle y \,\middle\|\, S_2^{\phi}\left[\langle x \parallel S^{\phi}[\,]\rangle\right] \right\rangle \\
c_1^{\phi}[c^{\bar{\phi}}[c_2^{\phi}[\,]]] & \rhd_{\mathrm{GP}} & c_1^{\phi}[c_2^{\phi}[c^{\bar{\phi}}[\,]]]
\end{array}$$

We assume commutativity and independence of the reordered command contexts.

Figure 10 – Focused equivalences

Appendix A
Phase Normal Forms

Inspired by the work on multi-focusing [9], in Figure 10 we split $\simeq_{\mathrm{P}}$ into two sub-relations: a *local* phase equivalence $\simeq_{\mathrm{LP}}$, *that* permutes focused values and stacks inside a single phase, and a *global* phase reordering $\simeq_{\mathrm{GP}}$. The global reordering is orientable into a rewrite relation $\rhd_{\mathrm{GP}}$, which gives a notion of canonical forms at the level of whole phases—this only gives *quasi*-normal forms on the term syntax since the (decidable) relation $\simeq_{\mathrm{LP}}$ has no natural ordering.

The global rewrite lets a focused commutative context of phase ϕ (which may be only a fragment of a larger context) jump across another context of opposite phase $\bar{\phi}$. Note that it must jump into another context of the same phase, and thus cannot split the $\bar{\phi}$-context in two—we say that a focused context inside a focused command is *maximal* if its the surrounding contexts are of the opposite phase. This restriction is necessary for the rewrite to be terminating: with the orientation of the reduction and this restriction, the outermost maximal contexts grows bigger at each rewrite step.

Lemma 22. *$\rhd_{\mathrm{GP}}$ is terminating.*

Note that the reordered $c^{\bar{\phi}}$ may be duplicated if c_2 has several holes. The termination order is thus the tree of sizes of maximal contexts; the tree with the biggest element coming first is the smaller. This is well-ordered if the total size of trees arising from the rewrite of a given element is bounded. The possible rewrites of a term of size N are bounded in size by 2^N (an impossible worst case where all elements are duplicating and get rewritten in a such a way that they do duplicate the rest of the term).

The combination of $\simeq_{\mathrm{LP}}$ and $\rhd_{\mathrm{GP}}$ subsumes $\simeq_{\mathrm{P}}$:

Lemma 23. *$f_1 \simeq_{\mathrm{P}} f_2$ if and only if $f_1 \rhd_{\mathrm{GP}}^{*} f_1'$ and $f_2 \rhd_{\mathrm{GP}}^{*} f_2'$ with $f_1' \simeq_{\mathrm{LP}} f_2'$.*

The equivalence $\simeq_{\mathrm{LP}}$ is decidable and local, so it is natural to manipulate terms up to this equivalence. One way to do this is to introduce an explicit *multi-focusing* syntax that performs a series of independent plays simultaneously on different variables and optionally on $\star$; this corresponds to a multi-let syntax ($\mathsf{let}\,\bar{x}\,\mathsf{be}\,\bar{e}\,\mathsf{in}\,t$) in some calculi, or to the multi-case pattern-matching construct of Altenkirch, Dybjer, Hofmann, and Scott [3]. Another equivalent choice, which we adopt for simplicity, is to pick an arbitrary order for local phases.

Definition 24. We define $\lfloor f \rfloor_{\mathrm{LP}}$ as an arbitrary choice of representative in the $\simeq_{\mathrm{LP}}$-equivalence class of the focused command f. We will also write $\lfloor f \rfloor_{\mathrm{GP}}$ for the unique $\rightarrow_{\mathrm{GP}}$-normal form of f, and $\lfloor f \rfloor_{\mathrm{P}}$ for $\lfloor \lfloor f \rfloor_{\mathrm{GP}} \rfloor_{\mathrm{LP}}$. Finally, if c is SN, and its normal form has the focused form f, we will write $\lfloor c \rfloor_{\mathrm{P}}$ for $\lfloor f \rfloor_{\mathrm{P}}$.

Appendix B
Proofs from Section IV

Proof of Lemma 6: We deterministically rewrite any normal command c into an equivalent focused command $\mathrm{F_c}[\![c]\!]$ as follows:

$$\begin{aligned}
\mathrm{F_c}[\![\langle V_+ \parallel \star\rangle]\!] &\stackrel{\text{def}}{=} \langle \mathrm{F_{V_+}}[\![V_+]\!] \parallel \star\rangle \\
\mathrm{F_c}[\![\langle x^+ \parallel e_+\rangle]\!] &\stackrel{\text{def}}{=} \langle x^+ \parallel \mathrm{F_{e_+}}[\![e_+]\!]\rangle \\
\mathrm{F_c}[\![\langle x^{\ominus} \parallel S_{\ominus}\rangle]\!] &\stackrel{\text{def}}{=} \langle x^{\ominus} \parallel \mathrm{F_{S_{\ominus}}}[\![S_{\ominus}]\!]\rangle \\
\mathrm{F_c}[\![\langle t_{\ominus} \parallel \star\rangle]\!] &\stackrel{\text{def}}{=} \langle \mathrm{F_{t_{\ominus}}}[\![t^{\ominus}]\!] \parallel \star\rangle \\
\mathrm{F_{V_+}}[\![x^+]\!] &\stackrel{\text{def}}{=} x^+ \\
\mathrm{F_{V_+}}[\![(V_1, V_2)]\!] &\stackrel{\text{def}}{=} (\mathrm{F_V}[\![V_1]\!], \mathrm{F_V}[\![V_2]\!]) \\
\mathrm{F_{V_+}}[\![\iota_1(V)]\!] &\stackrel{\text{def}}{=} \iota_1(\mathrm{F_V}[\![V]\!]) \\
\mathrm{F_V}[\![V_+]\!] &\stackrel{\text{def}}{=} \mathrm{F_{V_+}}[\![V_+]\!] \\
\mathrm{F_V}[\![\mu\,\star^{\ominus}.c]\!] &\stackrel{\text{def}}{=} \mu\,\star^{\ominus}.\mathrm{F_c}[\![c]\!] \\
\mathrm{F_V}[\![t_{\ominus}]\!] &\stackrel{\text{def}}{=} \mu\,\star^{\ominus}.\langle \mathrm{F_{t_{\ominus}}}[\![t_{\ominus}]\!] \parallel \star\rangle \\
\mathrm{F_{t_{\ominus}}}[\![x^{\ominus}]\!] &\stackrel{\text{def}}{=} x^{\ominus} \\
\mathrm{F_{t_{\ominus}}}[\![\mu(x \cdot \star).c]\!] &\stackrel{\text{def}}{=} \mu(x \cdot \star).\mathrm{F_c}[\![c]\!] \\
\mathrm{F_{t_{\ominus}}}[\![\mu{<}\star.c\,;\star.c'{>}]\!] &\stackrel{\text{def}}{=} \mu{<}\star.\mathrm{F_c}[\![c]\!]\,;\star.\mathrm{F_c}[\![c']\!]{>} \\
\mathrm{F_{S_{\ominus}}}[\![\star]\!] &\stackrel{\text{def}}{=} \star \\
\mathrm{F_{S_{\ominus}}}[\![V \cdot S]\!] &\stackrel{\text{def}}{=} \mathrm{F_V}[\![V]\!] \cdot \mathrm{F_S}[\![S]\!] \\
\mathrm{F_{S_{\ominus}}}[\![\pi_i \cdot S]\!] &\stackrel{\text{def}}{=} \pi_i \cdot \mathrm{F_S}[\![S]\!] \\
\mathrm{F_S}[\![\tilde{\mu}x^+.c]\!] &\stackrel{\text{def}}{=} \tilde{\mu}x.\mathrm{F_c}[\![c]\!] \\
\mathrm{F_S}[\![e_+]\!] &\stackrel{\text{def}}{=} \tilde{\mu}x.\langle \star \parallel \mathrm{F_{e_+}}[\![e_+]\!]\rangle \\
\mathrm{F_{e_+}}[\![\star]\!] &\stackrel{\text{def}}{=} \star \\
\mathrm{F_{e_+}}[\![\tilde{\mu}[x_1.c_1 \,|\, x_2.c_2]]\!] &\stackrel{\text{def}}{=} \tilde{\mu}[x_1.\mathrm{F_c}[\![c_1]\!] \,|\, x_2.\mathrm{F_c}[\![c_2]\!]] \\
\mathrm{F_{e_+}}[\![\tilde{\mu}(x, y).c]\!] &\stackrel{\text{def}}{=} \tilde{\mu}(x, y).\mathrm{F_c}[\![c]\!]
\end{aligned}$$

In words, $\mathrm{F_c}[\![c]\!]$ is obtained from c by applying an expansion $E\mu$ at every phase change from V_+ to $t_{\ominus}$ and an expansion $E\tilde{\mu}$ at every phase change from $S_{\ominus}$ to e_+. Note that the two ways to translate $\langle x^+ \parallel \star\rangle$, as either $\langle V_+ \parallel \star\rangle$ or $\langle x^+ \parallel e_+\rangle$, give the same translation $\langle x^+ \parallel \star\rangle$. ∎

Proof of Lemma 7: By induction on the derivation of algorithmic equivalence. For example in the $\mu(x, y)$ case, we need to prove that $\langle x^+ \parallel \mu(x_1, x_2).c\rangle \simeq_{\mathrm{RE}} c'$. Let us remark that we have, for any command c'':

$$\begin{aligned}
c'' &\lhd_{\mathrm{R}_{\tilde{\mu}}} \langle x^+ \parallel \tilde{\mu}x^+.c''\rangle \\
&\simeq_{\mathrm{E}_{\otimes}} \left\langle x^+ \,\middle\|\, \mu(x_1, x_2).\langle (x_1, x_2) \parallel \tilde{\mu}x^+.c''\rangle \right\rangle \\
&\rhd_{\mathrm{R}_{\tilde{\mu}}} \langle x^+ \parallel \mu(x_1, x_2).c''[(x_1, x_2)/x^+]\rangle
\end{aligned}$$

If we use this property with $c'' = c'$, we obtain:

$$c' \simeq_{\mathrm{RE}} \langle x^+ \parallel \mu(x_1, x_2).c'[(x_1, x_2)/x^+]\rangle$$

If we use it with $\langle x^+ \parallel \mu(x_1, x_2).c\rangle$ as a whole, we get:

$$\begin{aligned}
&\langle x^+ \parallel \mu(x_1, x_2).c\rangle \\
&\simeq_{\mathrm{RE}} \left\langle x^+ \,\middle\|\, \mu(x_1, x_2).\langle x^+ \parallel \mu(x_1, x_2).c\rangle[(x_1, x_2)/x^+] \right\rangle \\
&= \left\langle x^+ \,\middle\|\, \mu(x_1, x_2).\langle (x_1, x_2) \parallel \mu(x_1, x_2).c[(x_1, x_2)/x^+]\rangle \right\rangle \\
&\rightarrow_{\mathrm{R}_{\otimes}} \langle x^+ \parallel \mu(x_1, x_2).c[(x_1, x_2)/x^+]\rangle
\end{aligned}$$

Then, to show that $\langle x^+ \parallel \mu(x_1, x_2).c[(x_1, x_2)/x^+]\rangle$ and $\langle x^+ \parallel \mu(x_1, x_2).c'[(x_1, x_2)/x^+]\rangle$ are equivalent, it suffices to show that $c[(x_1, x_2)/x^+]$ and $c'[(x_1, x_2)/x^+]$ are equivalent, and this is exactly our induction hypothesis. ∎

Proof of Lemma 8: The proof is as for the non-extended version. We used as an intermediate lemma the fact that, for any c, we have $c \simeq_{\mathrm{RE}} \langle x^+ \| \mu(x_1,x_2).c[(x_1,x_2)/x^+]\rangle$. We just need to extend this result to prove $c \simeq_{\mathrm{RE}} \langle V_+ \| \mu(x_1,x_2).c[(x_1,x_2)/V_+]\rangle$:

$$\begin{aligned} c &\leftarrow_{\mathrm{R}_{\tilde\mu}} \langle V_+ \| \tilde\mu x^+.c[x^+/V_+]\rangle \\ &\simeq_{\mathrm{E}_\oplus} \left\langle V_+ \,\middle\|\, \mu(x_1,x_2).\langle (x_1,x_2) \| \tilde\mu x^+.c[x^+/V_+]\rangle \right\rangle \\ &\to_{\mathrm{R}_\times} \langle V_+ \| \mu(x_1,x_2).c[(x_1,x_2)/V_+]\rangle \end{aligned}$$

(Note that $c[x^+/V_+][(x_1,x_2)/x^+] = c[(x_1,x_2)/V_+]$ because x is assumed fresh in c by the Barendregt convention.) ∎

Proof of Lemma 9: By induction. To justify the induction we need an induction measure that is strictly decreasing on, for example,

$$\frac{c[(x_1,x_2)/V_+] \equiv_{\mathrm{A}} \langle (x_1,x_2) \| \mu(x_1,x_2).c[(x_1,x_2)/V_+]\rangle}{\langle V_+ \| \mu(x_1,x_2).c\rangle \equiv_{\mathrm{A}_N} \langle V_+ \| \mu(x_1,x_2).c\rangle}$$

As in the article of Abel and Coquand and in Goguen [30] which they refer to, we simply need an induction measure that favours abstractors over constructors, e.g.:

$$\begin{aligned} |\langle V \| \tilde\mu q.c\rangle| &:= 2.|c| \\ |\langle \mu q.c \| S\rangle| &:= 2.|c| \\ |\langle V[c_i] \| S[c_j]\rangle| &:= \Sigma_i |c_i| + \Sigma_j |c_j| + 1 \end{aligned}$$

Notice then that the sum of the measures of the premises is strictly inferior to the sum of the measures of its conclusion, justifying building such derivations by induction. ∎

To facilitate proofs traversing abstractors and performing substitution on their bindings, we use a generic notation $\mu q.c$ or $\tilde\mu q.c$, where q is a binding pattern. Because such patterns are also valid expressions or contexts (and are meaningful in c, as the variables bound in $\mu q.$ scope over c) we can write $c[q/V]$ or $c[q/S]$, replacing all occurences of V (or S) by q. There is a subtlety in the case of the multi-command abstractors: $\mu\!<\!\star.c_1\,;\star.c_2\!>$ and $\tilde\mu[x_1.c_1|x_2.c_2]$, respectively: in that case the c in the general syntax $\mu q.c$ or $\tilde\mu q.c$ denotes the pair (c_1,c_2), and the substitutions $c[\,q/S\,]$ and $c[\,q/V\,]$ respectively denote the pairs $(c_1[\,\pi_1\cdot\star/S\,], c_2[\,\pi_2\cdot\star/S\,])$ and $(c_1[\,\iota_1(x_1)/V\,], c_2[\,\iota_2(x_2)/V\,])$. Statements on such pairs should be lifted as conjunctions; for exmaple,the equation $c[q/S] = c'[q/S]$ (for some command c') should be understood as the conjunction of $c_1[\pi_1\cdot\star/S] = c'[\pi_1\cdot\star/S]$ and $c_2[\pi_2\cdot\star/S] = c_2'[\pi_2\cdot\star/S]$. As an example of its usefulness, this notation allows a very compact reformulation of the abstractor rules in the definition of extended algorithmic equivalence 8b:

$$\frac{c[q/S] \equiv_{\mathrm{A}+} c'[q/S]}{\langle \mu q.c \| S\rangle \equiv_{\mathrm{A}+_N} c'}\ \text{A-}\mu q \qquad \frac{c[q/V] \equiv_{\mathrm{A}+} c'[q/V]}{\langle V \| \tilde\mu q.c\rangle \equiv_{\mathrm{A}+_N} c'}\ \text{A-}\tilde\mu q$$

To prove transitivity of the extended algorithmic equivalence, we first need a lemma. Let us write $\lfloor c \rfloor_{\mathrm{R}}$ for the $\to_{\mathrm{R}}$-normal form of c.

Lemma 25. *Any derivation of $\lfloor c_1\rfloor_R \equiv_{\mathrm{A}+_N} \lfloor c_2\rfloor_R$ can be turned into a derivation of $\lfloor c_1[q/V_+]\rfloor_R \equiv_{\mathrm{A}+_N} \lfloor c_2[q/V_+]\rfloor_R$ (resp. $\lfloor c_1[q/S_\ominus]\rfloor_R \equiv_{\mathrm{A}+_N} \lfloor c_2[q/S_\ominus]\rfloor_R$) of equal or smaller height when q does not appear in c_1, c_2.*

Proof: The proof goes by induction on the $(\equiv_{\mathrm{A}+_N})$-derivation, with a strengthened induction hypothesis: for any substitution σ, pattern q and value V_+ (respectively stack $S_\ominus$), any derivation of $\lfloor c_1[\sigma]\rfloor_R \equiv_{\mathrm{A}+_N} \lfloor c_2[\sigma]\rfloor_R$ can be turned into a derivation of $\lfloor c_1[q/V_+][\sigma]\rfloor_R \equiv_{\mathrm{A}+_N} \lfloor c_2[q/V_+][\sigma]\rfloor_R$ (respectively $\lfloor c_1[q/S_\ominus][\sigma]\rfloor_R \equiv_{\mathrm{A}+_N} \lfloor c_2[q/S_\ominus][\sigma]\rfloor_R$) of equal or smaller length, when q does not appear in c_1, c_2 or σ.

Let us assume, for instance, that the last rules of our assumption are of the following form:

$$\frac{\lfloor c_1\sigma[q_1/V_1[\sigma]]\rfloor_R \equiv_{\mathrm{A}+_N} \lfloor c_2\sigma[q_1/V_1[\sigma]]\rfloor_R}{\dfrac{c_1[\sigma][q_1/V_1[\sigma]] \equiv_{\mathrm{A}+} c_2[\sigma][q_1/V_1[\sigma]]}{\langle V_1[\sigma] \| \tilde\mu q_1.c_1[\sigma]\rangle \equiv_{\mathrm{A}+_N} c_2[\sigma]}}$$

We have to find a derivation for $\lfloor\langle V_1 \| \tilde\mu q_1.c_1\rangle[q/V][\sigma]\rfloor_R \equiv_{\mathrm{A}+_N} \lfloor c_2[q/V][\sigma]\rfloor_R$ of equal and smaller height. There are two possible cases for the structure of $\lfloor\langle V_1 \| \tilde\mu q_1.c_1\rangle[q/V][\sigma]\rfloor_R$ that is $\lfloor\langle V_1[q/V][\sigma] \| \tilde\mu q_1.c_1[q/V][\sigma]\rangle\rfloor_R$: either the constructor $V_1[q/V][\sigma]$ does not match the abstractor $\tilde\mu q_1$, and the $\to_{\mathrm{R}}$-normal form is just $\langle V_1[q/V][\sigma] \| \tilde\mu q_1.\lfloor c_1[q/V][\sigma]\rfloor_R\rangle$, or it does, and we have a head $\rhd_{\mathrm{R}}$-reduction to perform.

In the first case, let us write V_1' for $\lfloor V_1[q/V]\rfloor_R$. We can perform the following first steps of inference (with the top sequent still to be proved).

$$\frac{\left\lfloor\lfloor c_1[q/V][\sigma]\rfloor_R[q_1/V_1'[\sigma]]\right\rfloor_R \equiv_{\mathrm{A}+_N} \left\lfloor\lfloor c_2[q/V][\sigma]\rfloor_R[q_1/V_1'[\sigma]]\right\rfloor_R}{\dfrac{\lfloor c_1[q/V][\sigma]\rfloor_R[q_1/V_1'[\sigma]] \equiv_{\mathrm{A}+} \lfloor c_2[q/V][\sigma]\rfloor_R[q_1/V_1'[\sigma]]}{\langle V_1'[\sigma] \| \tilde\mu q_1.\lfloor c_1[q/V][\sigma]\rfloor_R\rangle \equiv_{\mathrm{A}+_N} \lfloor c_2[q/V][\sigma]\rfloor_R}}$$

Notice that $\left\lfloor\lfloor c_1[q/V][\sigma]\rfloor_R[q_1/V_1'[\sigma]]\right\rfloor_R$ is equal to $\left\lfloor\lfloor c_1[q/V][\sigma]\rfloor_R[q_1/V_1'[\sigma]]\right\rfloor_R$. The top sequent is then proved by induction hypothesis, using the larger substitution $[\sigma][q_1/V_1'[\sigma]]$.

In the second case, we assume that $V_1[q/V][\sigma]$ starts with the same head constructor as the pattern q_1. But we assumed that $\langle V_1 \| \tilde\mu q_1.c_1\rangle[\sigma]$ was a normal form, so we know that $V_1[\sigma]$ and q_1 do not match. If $V_1[\sigma]$ does not match q_1 but $V_1[q/V][\sigma]$ does, the matching part is either brought by q (this means that the occurrence of V in V_1 is at the top) or the substitution of $V_1[q/V]$ by σ. But the latter is impossible, given that we assumed that q does not appear in σ. So we have that $V_1 = V$, and $V_1[q/V][\sigma] = V[q/V][\sigma] = q[\sigma] = q$. In particular, we can always α-convert q_1 to be equal to q, and then we can prove our goal:

$$\lfloor\langle V_1 \| \tilde\mu q_1.c_1\rangle[q/V][\sigma]\rfloor_R \equiv_{\mathrm{A}+_N} \lfloor c_2[q/V][\sigma]\rfloor_R$$

from our assumption:

$$\lfloor c_1[\sigma][q_1/V_1[\sigma]]\rfloor_R \equiv_{\mathrm{A}+_N} \lfloor c_2[\sigma][q_1/V_1[\sigma]]\rfloor_R$$

with the following equational reasoning:

$$\begin{aligned} & \lfloor\langle V_1 \| \tilde\mu q_1.c_1\rangle[q/V][\sigma]\rfloor_R \equiv_{\mathrm{A}+_N} \lfloor c_2[q/V][\sigma]\rfloor_R \\ \Leftrightarrow\ & \lfloor\langle V_1[q/V] \| \tilde\mu q_1.c_1[q/V]\rangle[\sigma]\rfloor_R \equiv_{\mathrm{A}+_N} \lfloor c_2[q/V][\sigma]\rfloor_R \\ \Leftrightarrow\ & (V_1 = V) \\ & \lfloor\langle q \| \tilde\mu q_1.c_1[q/V]\rangle[\sigma]\rfloor_R \equiv_{\mathrm{A}+_N} \lfloor c_2[q/V][\sigma]\rfloor_R \\ \Leftrightarrow\ & (q \notin \sigma) \\ & \lfloor\langle q[\sigma] \| \tilde\mu q_1.c_1[q/V][\sigma]\rangle\rfloor_R \equiv_{\mathrm{A}+_N} \lfloor c_2[q/V][\sigma]\rfloor_R \\ \Leftrightarrow\ & (q_1 = q) \\ & \lfloor\langle q \| \tilde\mu q_1.c_1[q/V][\sigma]\rangle\rfloor_R \equiv_{\mathrm{A}+_N} \lfloor c_2[q/V][\sigma]\rfloor_R \\ \Leftrightarrow\ & \lfloor\langle q_1 \| \tilde\mu q_1.c_1[q/V][\sigma]\rangle\rfloor_R \equiv_{\mathrm{A}+_N} \lfloor c_2[q/V][\sigma]\rfloor_R \\ \Leftrightarrow\ & (\rhd_{\mathrm{R}}) \\ & \lfloor c_1[q/V][\sigma]\rfloor_R \equiv_{\mathrm{A}+_N} \lfloor c_2[q/V][\sigma]\rfloor_R \\ \Leftrightarrow\ & (q \notin \sigma) \\ & \lfloor c_1[\sigma][q/V[\sigma]]\rfloor_R \equiv_{\mathrm{A}+_N} \lfloor c_2[\sigma][q/V[\sigma]]\rfloor_R \\ \Leftrightarrow\ & (q_1 = q \text{ and } V_1 = V) \\ & \lfloor c_1[q_1/V_1][\sigma]\rfloor_R \equiv_{\mathrm{A}+_N} \lfloor c_2[q_1/V_1][\sigma]\rfloor_R \end{aligned}$$

∎

Proof of Lemma 10: Combining $c_1 \equiv_{\mathrm{A}} c_2$ and $c_2 \equiv_{\mathrm{A}} c_3$ gives the following series of relations: $c_1 \to_{\mathrm{R}} c_1' \equiv_{\mathrm{A}_N} c_2' \leftarrow_{\mathrm{R}} c_2 \to_{\mathrm{R}} c_2'' \equiv_{\mathrm{A}_N} c_3' \leftarrow_{\mathrm{R}} c_3$. By confluence of $(\to_{\mathrm{R}})$ we know that $c_2' = c_2''$ (they are both normal forms), so we have to prove transitivity with $c_1' \equiv_{\mathrm{A}_N} c_2' \equiv_{\mathrm{A}_N} c_3'$. We need to inspect both inference rules; the easy cases are when c_2' starts with an abstractor which guides the equivalence on both sides. The more interesting cases arise when the abstractors guiding the equivalence are in c_1' or c_3'. We detail one of those cases. Suppose we have:

$$\frac{c_1[q_1/V_1] \equiv_{\mathrm{A}} c_2[q_1/V_1]}{\langle V_1 \parallel \tilde{\mu} q_1.c_1\rangle \equiv_{\mathrm{A}_N} c_2} \qquad \frac{c_2[q_3/S_3] \equiv_{\mathrm{A}} c_3[q_3/S_3]}{c_2 \equiv_{\mathrm{A}_N} \langle \mu q_3.c_3 \parallel S_3\rangle}$$

Let us take $S_3' := S_3[q_1/V_1]$. We build a derivation of the form:

$$\frac{\frac{\frac{\lfloor c_1[q_1/V_1]\rfloor_{\mathrm{R}}[q_3/\lfloor S_3'\rfloor_{\mathrm{R}}] \equiv_{\mathrm{A}} \lfloor c_3[q_1/V_1]\rfloor_{\mathrm{R}}[q_3/\lfloor S_3'\rfloor_{\mathrm{R}}]}{\lfloor c_1[q_1/V_1]\rfloor_{\mathrm{R}} \equiv_{\mathrm{A}+_N} \lfloor\langle \mu q_3.c_3[q_1/V_1] \parallel S_3'\rangle\rfloor_{\mathrm{R}}}}{c_1[q_1/V_1] \equiv_{\mathrm{A}+} \langle \mu q_3.c_3[q_1/V_1] \parallel S_3'\rangle}}{\langle V_1 \parallel \tilde{\mu} q_1.c_1\rangle \equiv_{\mathrm{A}+_N} \langle \mu q_3.c_3 \parallel S_3\rangle}$$

To conclude our proof, we need to prove the premise $c_1[\,q_1/V_1\,][\,q_3/S_3'\,] \equiv_{\mathrm{A}} c_3[\,q_1/V_1\,][\,q_3/S_3'\,]$ (we omit the $\to_{\mathrm{R}}$-normalization marks for readability of the notation). We build it by induction hypothesis, using transitivity on two $(\equiv_{\mathrm{A}})$ derivations of smaller height. The first is $c_1[q_1/V_1][q_3/S_3'] \equiv_{\mathrm{A}} c_2[q_1/V_1][q_3/S_3']$, obtained from our hypothesis $c_1[q_1/V_1] \equiv_{\mathrm{A}} c_2[q_1/V_1]$ by applying the previous lemma—it is thus strictly smaller than the derivation of $\langle V_1 \parallel \tilde{\mu} q_1.c_1\rangle \equiv_{\mathrm{A}_N} c_2$ we had as hypothesis. The second, $c_2[q_1/V_1][q_3/S_3'] \equiv_{\mathrm{A}} c_3[q_1/V_1][q_3/S_3']$, requires applying the technical lemma on our assumption $c_2 \equiv_{\mathrm{A}_N} \langle\, \mu q_3.c_3 \parallel S_3 \,\rangle$, to get a non-larger derivation of $c_2[\,q_1/V_1\,] \equiv_{\mathrm{A}} \langle\, \mu q_3.c_3[q_1/V_1] \parallel S_3' \,\rangle$; inspecting this derivation, which starts with the same structure as the one for $c_2 \equiv_{\mathrm{A}_N} \langle\, \mu q_3.c_3 \parallel S_3 \,\rangle$, gives a non-larger derivation of $c_2[q_1/V_1][q_3/S_3'] \equiv_{\mathrm{A}} c_3[q_1/V_1][q_3/S_3']$. ∎

Proof of Lemma 11: The proof goes by proving each equation of $(\simeq_{\mathrm{RE}_{SN}})$ admissible for $(\equiv_{\mathrm{A}})$. This is immediate for $(\to_{\mathrm{R}})$-reductions thanks to the rule A-R. For the expansions, we can assume that both sides of the expansion are in $(\to_{\mathrm{R}})$-normal form—this is where we need $(\simeq^{\mathrm{SN}}_{\mathrm{RE}})$ rather than $(\simeq_{\mathrm{RE}})$, as otherwise a strongly-normalizable command could be $(\to_{\mathrm{E}})$-expanded into a non-normalizing one. We have for example the case $\langle x^+ \parallel e_+\rangle \to_{\mathrm{E}_\otimes} \left\langle x^+ \middle\| \mu(x_1,x_2).\langle (x_1,x_2) \parallel e_+\rangle \right\rangle$ which is proved by the following inference.

$$\frac{\langle (x_1,x_2) \parallel e_+\rangle[(x_1,x_2)/x^+] = \langle x_+ \parallel e_+\rangle[(x_1,x_2)/x^+]}{\left\langle x^+ \middle\| \mu(x_1,x_2).\langle (x_1,x_2) \parallel e_+\rangle \right\rangle \equiv_{\mathrm{A}} \langle x^+ \parallel e_+\rangle}$$

Note that we do not need to handle the case of the extensions $E\mu$ and $E\tilde{\mu}$, only those that inspect value or stack constructors. Indeed, putting a normal form into focused form, as described in Lemma 6, normalizes the $E\mu, \tilde{\mu}$ structure: we have $f \rhd_{E\mu,\tilde{\mu}} f'$ only if $f = f'$. ∎

Proof of Lemma 15: Consider for example the case $c_1 = \langle x^+ \parallel \tilde{\mu}[z_1.[\,]_1 | z_2.[\,]_2]\rangle$. Given an arbitrary c_2, let us define:

$$e_+ \stackrel{\text{def}}{=} \tilde{\mu} x^+.c_2[c_1[\,]] = \tilde{\mu} x^+.c_2\left[\langle x^+ \parallel \tilde{\mu}[z_1.[\,]_1 | z_2.[\,]_2]\rangle\right]$$

Then we have:

$$\begin{aligned}
&c_2[c_1[\,]]\\
&\lhd_{\mathrm{R}_{\tilde{\mu}}} \langle x^+ \parallel e_+\rangle\\
&\to_{\mathrm{E}_\oplus} \left\langle x^+ \middle\| \tilde{\mu}\left[z_1.\langle \iota_1(z_1) \parallel e_+\rangle \middle| z_2.\langle \iota_2(z_2) \parallel e_+\rangle\right] \right\rangle\\
&\to^*_{\mathrm{R}_{\tilde{\mu},\oplus}} \langle x^+ \parallel \tilde{\mu}[z_1.c_2[\,]_1 | z_1.c_2[\,]_2]\rangle\\
&= c_1[c_2[\,]]
\end{aligned}$$

This concludes the proof in the case $c_1 = \langle x^+ \parallel \tilde{\mu}[z_1.[\,]|z_2.[\,]]\rangle$. The other cases are rather similar. For example, if $c_1 = \langle x^+ \parallel \tilde{\mu}(y_1,y_2).[\,]\rangle$, then again with $e_+ := \tilde{\mu} x^+.c_2[c_1[\,]]$ we have:

$$\begin{aligned}
&c_2[c_1[\,]]\\
&\lhd_{\mathrm{R}_{\tilde{\mu}}} \langle x^+ \parallel e_+\rangle\\
&\to_{\mathrm{E}_\oplus} \left\langle x^+ \middle\| \tilde{\mu}(y_1,y_2).\langle (y_1,y_2) \parallel e_+\rangle \right\rangle\\
&\to_{\mathrm{R}_{\tilde{\mu}}} \left\langle x^+ \middle\| \tilde{\mu}(y_1,y_2).c_2\left[\langle (y_1,y_2) \parallel \tilde{\mu}(y_1,y_2).[\,]\rangle\right] \right\rangle\\
&\to^*_{\mathrm{R}_\otimes} \langle x^+ \parallel \tilde{\mu}(y_1,y_2).c_2[\,]\rangle\\
&= c_1[c_2[\,]]
\end{aligned}$$
∎

Proof of Lemma 16: The reduction rule (β) is very strong, so this proof is easier and more direct than the previous one.

Notice first that while abstractor commutative contexts may have several holes, consturctor commutative contexts always have exactly one hole. Furthermore, a constructor context $c_1[\,]$ can only take four possible shapes. It is of either forms $\langle\, x^\ominus \parallel S[\,]\,\rangle$ or $\langle\, V[\,] \parallel \star\,\rangle$, and the hole $[\,]$ is after an abstractor of either forms $\mu^\ominus\star.[\,]$ or $\tilde{\mu} x^+.[\,]$—the two other possible abstractors are not values. Now consider two independent commutative contexts c_1 and c_2 composed to a command c, that is $c_1[\,c_2[c]\,]$. It is impossible that both commutative contexts end with a $\mu^+\star.[\,]$, as they would not be independent. At least one of them must then end with an abstractor of the form $\tilde{\mu} x^+.[\,]$. We will show that commuting this command with the other is sound with respect to $(\simeq^{\mathrm{SN}}_{\mathrm{RE}\beta})$. Without loss of generality (the result is symmetric), we can assume it is c_1 that is of the form $c_1'(\tilde{\mu} x^+.[\,])$, and show that we have $c_1'(\tilde{\mu} x^+.c_2[\,c\,]) \simeq^{\mathrm{SN}}_{\mathrm{RE}\beta} c_2[\,c_1'(\tilde{\mu} x^+.c)\,]$. Remark that our independence assumption tells us that x^+ is not bound in c_2.

$$\begin{aligned}
&c_1'(\tilde{\mu} x^+.c_2[c])\\
&\lhd_{\mathrm{R}_\mu} \langle \mu^+\star.c_1'(\star) \parallel \tilde{\mu} x^+.c_2[c]\rangle\\
&\rhd_\beta c_2[c[\mu^+\star.c_1'(\star)/^f x^+]]\\
&\leftarrow_\beta c_2\left[\langle \mu^+\star.c_1'(\star) \parallel \tilde{\mu} x^+.c\rangle\right]\\
&\to_{\mathrm{R}_\mu} c_2[c_1'(\tilde{\mu} x^+.c)]
\end{aligned}$$
∎

Proof of Proposition 20: We show that $(\simeq_{\mathrm{M}}) \subseteq (\simeq^{\mathrm{SN}}_{\mathrm{RE}\beta})$ and $(\simeq_{\mathrm{W}}) \subseteq (\simeq^{\mathrm{SN}}_{\mathrm{RE}\beta})$ separately.

For $(\simeq_{\mathrm{W}}) \subseteq (\simeq^{\mathrm{SN}}_{\mathrm{RE}\beta})$ we have to show that if $c[\,]$ is a commutative context and f uses no variable bound by $c[\,]$ over its hole(s), then $c[f] \simeq^{\mathrm{SN}}_{\mathrm{RE}\beta} f$. By case analysis on commutative contexts, we see that $c[\,]$ is necessarily of the form $c'[\tilde{\mu} q^+.[]]$ (c cannot bind $\star$ as f necessarily uses it). We thus have that $c[f] = c'[\tilde{\mu} q^+.f] \simeq_{\mathrm{R}\mu} \left\langle \mu^+\star.c'[\star] \middle\| \tilde{\mu} x^+.\langle x^+ \parallel \tilde{\mu} q^+.f\rangle \right\rangle$ for a fresh x. We now prove that, if f does not use q, then $\tilde{\mu} q^+.f \simeq_{\mathrm{RE}} \tilde{\mu} y^+.f$ for a fresh y; this allows to conclude with $\left\langle \mu^+\star.c'[\star] \middle\| \tilde{\mu} x^+.\langle x^+ \parallel \tilde{\mu} q^+.f\rangle \right\rangle \simeq^{\mathrm{SN}}_{\mathrm{RE}} \langle \mu^+\star.c'[\star] \parallel \tilde{\mu} y^+.f\rangle \rhd_\beta f$.

If $\tilde{\mu} q^+.f$ is of the form $\tilde{\mu} z^+.f$ for some variable z, the result is immediate by α-conversion.

If $\tilde{\mu} q^+.f$ is of the form $\tilde{\mu}(z_1,z_2).f$ where f does not use z_1 or z_2, then we have $\tilde{\mu}(z_1,z_2).f \simeq_{\mathrm{R}\tilde{\mu}} \tilde{\mu}(z_1,z_2).\langle (z_1,z_2) \parallel \tilde{\mu} y^+.f\rangle$ for some fresh y^+, and this is the $(E\otimes)$-expansion of $\tilde{\mu} y^+.f$.

If $\tilde{\mu}q^+.f$ is of the form $\tilde{\mu}[z.f\,|\,z.f]$ where f does not use z, we have that $\tilde{\mu}[z.f\,|\,z.f] \simeq_{\mathrm{R}} \tilde{\mu}\big[z.\langle \iota_1 z \parallel \tilde{\mu}y^+.f\rangle \big| z.\langle \iota_2 z \parallel \tilde{\mu}y^+.f\rangle\big]$ for some fresh y^+, and this is the $(E\oplus)$-expansion of $\tilde{\mu}y^+.f$.

For $(\simeq_{\mathrm{M}}) \subseteq (\simeq^{\mathrm{SN}}_{\mathrm{RE}\beta})$, we have to prove that for any pair of independent commutative contexts $c_1^\phi \# c_2^\phi$ and substitution ρ such that $c_1^\phi = c_2^\phi[\rho]$, we have $c_1^\phi[c_2^\phi[f]] \simeq^{\mathrm{SN}}_{\mathrm{RE}\beta} c_1^\phi[\rho[f]]$ for any focused command f. Again by case analysis on commutative contexts we see that c_1, c_2 cannot bind $\star$: a context that binds $\star$ over its hole must necessarily have a free occurrence of it, and thus cannot be independent with (an α-renaming of) itself. Thus c_1, c_2 are of the form $c'[\tilde{\mu}q_1^+.[\,]]$, $c'[\tilde{\mu}q_2^+.[\,]]$ for q_1 and q_2 α-equivalent through the variable-variable substitution ρ. We prove that $c'[\tilde{\mu}q_1^+.c'[\tilde{\mu}q_2^+.f]] \simeq^{\mathrm{SN}}_{\mathrm{RE}\beta} c'[\tilde{\mu}q_1^+.f[q_1/q_2]]$ as follows for some fresh $q =_\alpha q_1 =_\alpha q_2$:

$$
\begin{aligned}
&c'[\tilde{\mu}q_1^+.c'[\tilde{\mu}q_2^+.f]]\\
&\lhd_{\mathrm{R}\mu} \left\langle \mu^+\star.c'[\star] \,\middle\|\, \tilde{\mu}q_1^+.\langle \mu^+\star.c'[\star] \parallel \tilde{\mu}q_2^+.f\rangle \right\rangle\\
&\lhd_{\beta} \left\langle \mu^+\star.c'[\star] \,\middle\|\, \tilde{\mu}x^+.\left\langle x^+ \,\middle\|\, \tilde{\mu}q_1^+.\langle x^+ \parallel \tilde{\mu}q_2^+.f\rangle \right\rangle \right\rangle\\
&\simeq_{\mathrm{E}} \left\langle \mu^+\star.c'[\star] \,\middle\|\, \tilde{\mu}q^+.\left\langle q^+ \,\middle\|\, \tilde{\mu}x^+.\left\langle x^+ \,\middle\|\, \tilde{\mu}q_1^+.\langle x^+ \parallel \tilde{\mu}q_2^+.f\rangle \right\rangle \right\rangle \right\rangle\\
&\rhd_{\mathrm{R}\tilde{\mu}} \left\langle \mu^+\star.c'[\star] \,\middle\|\, \tilde{\mu}q^+.\left\langle q^+ \,\middle\|\, \tilde{\mu}q_1^+.\langle q^+ \parallel \tilde{\mu}q_2^+.f\rangle \right\rangle \right\rangle\\
&\simeq_{\mathrm{R}} \langle \mu^+\star.c'[\star] \parallel \tilde{\mu}q^+.f[q/q_1][q/q_2]\rangle\\
&=_\alpha \langle \mu^+\star.c'[\star] \parallel \tilde{\mu}q_1^+.f[q_1/q_2]\rangle\\
&\rhd_{\mathrm{R}\mu} c'[f[q_1/q_2]]
\end{aligned}
$$

∎

Proof of Theorem 21: We have to show that $\langle t_+ \parallel \tilde{\mu}x^+.c_0\rangle \simeq_{\mathrm{APMW}} c_0[t_+/^f x^+]$. The term t is either of the form $\mu^+\star.d_0$, or it is a value—in which case the result is immediate. Let us define c_1 as the focused normal form of c_0, and d_1 as the focused normal form of d_0. We also define as $d[\,]$ the command context, with holes expecting contexts, such that $d[\star] = d_1$ and $\star \notin \mathbf{fv}(d)$, and as $c[\,]$ the command context, with holes expecting expressions, such that $c[x^+] = c_1$ and $x^+ \notin \mathbf{fv}(c)$. We need to prove that $c[\mu^+\star.d[\star]] \simeq_{\mathrm{APM}} \langle \mu^+\star.d[\star] \parallel \tilde{\mu}x^+.c[x^+]\rangle$, the latter being $(\simeq^{\mathrm{SN}}_{\mathrm{RE}})$-equivalent to $d[\tilde{\mu}x^+.c[x^+]]$.

The main idea of the proof is that $c[\,\mu^+\star.[\,]\,]$ is *covered* by (compositions of) commutative contexts independent from $d[\,\tilde{\mu}x^+.[\,]\,]$: it can be described as a multi-hole context $c[\,\square_1\,][\,\square_2\,]\dots[\,\square_n\,]$ such that for each family of values $V_1, \dots, V_n$ and each family of holes $i \in [1; n]$, $c[V_1]\dots[V_{i-1}][\square_i][V_{i+1}]\dots[V_n]$ is a composition of commutative contexts so that:

$$
\begin{aligned}
&c[V_1]\dots[V_{i-1}]\big[\mu^+\star.d\big[\tilde{\mu}x^+.\langle x^+ \parallel \star\rangle\big]\big][V_{i+1}]\dots[V_n]\\
&\simeq_{\mathrm{P}} d\big[\tilde{\mu}x^+.c[V_1]\dots[V_{i-1}]\big[\mu^+\star.\langle x^+ \parallel \star\rangle\big][V_{i+1}]\dots[V_n]\big]
\end{aligned}
$$

Note that the commutation is possible because the contexts are independent. Indeed, the commutative contexts of $c[\,\mu^+\star.[\,]\,]$ neither use x^+ (by construction of $d[\,]$) nor bind it over its hole (any expression binder could be α-renamed), and $d[\tilde{\mu}x^+.]$ does not use $\star$ by construction, and cannot bind it over one of its hole, for $\star$ would then be shadowed in $d_1 = d[\star]$.

In the special case $n = 0$, $c[\,]$ is in fact a focused command c, and we have to show that $d[\tilde{\mu}x^+.c] \simeq_{\mathrm{APMW}} c$. This is immediate by weakening simplification $(\rhd_{\mathrm{W}})$, as x^+ is not used in c by construction.

Otherwise $(n \geq 1)$, by iteration on $[1; n]$ we can reorder n copies of $d[\mu x^+.[\,]]$ in front of c, obtaining the term:

$$
d\big[\tilde{\mu}x_1^+.\dots d\big[\tilde{\mu}x_n^+.c\big[\mu^+\star.\langle x_1 \parallel \star\rangle\big]\dots\big[\mu^+\star.\langle x_n \parallel \star\rangle\big]\big]\big]
$$

those copies can be merged, so this is $(\simeq_{\mathrm{M}})$-equivalent to $d[\tilde{\mu}x^+.c[x^+]]$ as desired.

Note that there is no need to interleave merging and phase reordering, as merging does not create new opportunities for phase reordering. It is essential, however, to apply merging on the maximally-reordered command, as reordering command creates merging opportunities.

We remark that there is a fundamental asymmetry in the proof technique that follows the asymmetry of the intuitionistic restriction: we permute, duplicate and reorder the value context $d[\tilde{\mu}x^+.[\,]]$ rather than the control context $c[\mu^+\star.[\,]]$, relying on a notion of "purity" that would be invalid in the full classical setting. ∎

Contents