Linear Lambda Calculus and Deep Inference

Luca Roversi

Dip. di Informatica, C.so Svizzera 185, 10149 Torino — Italy

Abstract. We introduce a deep inference logical system SBVr which extends SBV [6] with Rename, a self-dual atom-renaming operator. We prove that the cut free subsystem BVr of SBVr exists. We embed the terms of linear λ-calculus with explicit substitutions into formulas of SBVr. Our embedding recalls the one of full λ-calculus into π-calculus. The proof-search inside SBVr and BVr is complete with respect to the evaluation of linear λ-calculus with explicit substitutions. Instead, only soundness of proof-search in SBVr holds. Rename is crucial to let proof-search simulate the substitution of a linear λ-term for a variable in the course of linear β-reduction. Despite SBVr is a minimal extension of SBV its proof-search can compute all boolean functions, exactly like linear λ-calculus with explicit substitutions can do.

To Umberto Spina 1924 — 2000.

1 Introduction

We formalize unknown relations between basic functional computation and Deep inference (DI). We show that the computations of linear λ-calculus with explicit substitutions, at the core of functional programming, live as proof-search inside the logical system SBVr that we introduce, and which extends SBV, system at the core of DI [15].

Our motivation is to search how structural proof theory, based on DI methodology, can contribute to paradigmatic programming language design. We recall that logical systems designed under the prescriptions of DI allow inference rules to apply anywhere in a formula in the course of a derivation. This is in contrast to "shallow inference" of traditional structural proof theoretic formalisms, like natural deduction and sequent calculus, whose inference rules apply at the root of formulas and sequents only. Applying rules in depth implies that structural proof theory of a quite vast range of logics, i.e. classical, intuitionistic, linear and modal, has become very regular and modular. We expect that much regularity and modularity can highlight useful intrinsic properties and new primitives, or evaluation strategies, at the level of programs. The point is to look for the computational interpretation of derivations in DI style in the same vein as the one we are used to with shallow inference. We think that source of new programming primitives, or evaluation strategies, can be DI deductive systems whose inference rules only manipulate atoms of formulas, and for which new notions of proof normalization exist, in addition to cut elimination.

So far, [3] is the only work which focuses on the proposal of a Curry-Howard analogy for a DI system, and which we are aware of. It applies DI methodology to the natural deduction of the negative fragment of intuitionistic logic, and extracts an algebra of combinators from it where interpreting λ-terms.

Rather than intuitionistic logic, as in [3], we focus on SBV and its cut free subsystem BV. The point about BV is that, thanks to its "deepness", it naturally extends multiplicative linear logic (MLL) [5] with the non commutative binary operator Seq, denoted "$\lhd$". Second, from [4] we recall that Seq soundly and completely models the sequential communication of CCS concurrent and communicating systems [11], while Par ($\parr$) models parallel composition. Third, we recall that λ-calculus β-reduction incorporates a sequential behavior. We mean that the overall computation the redex $(\lambda x.M)\,N$ embodies can complete only *after* executing the β-reduction $(\lambda x.M)\,N \to_\beta M\{^N/_x\}$ that substitutes N for every occurrence of x in M, if any. The sequential behavior of β-reduction closely recalls the one in CCS. The difference is that, after consuming a prefix that takes the form of a λ-abstraction, β-reduction also requires to perform a substitution. So, if we map a λ-variable x into a BV formula, by a map $(\!|\cdot|\!)_\cdot$, as follows:

$$(\!|x|\!)_o = \langle x \lhd \overline{o}\rangle \tag{1}$$

we can start modeling the consumption of the λ-abstraction that takes place in the course of a β-reduction. Intuitively, x in (1) is the name of an input channel, to the left of Seq, eventually forwarded to the output channel o, associated to x by Seq. Definition (1) comes from the following embedding of sequents of *intuitionistic* multiplicative linear logic (IMLL) into SBV formulas:

$$(\alpha_1, \ldots, \alpha_m \vdash_{\mathsf{IMLL}} \beta)^\bullet = \langle(\overline{\alpha_1^\bullet} \otimes \cdots \otimes \overline{\alpha_m^\bullet}) \lhd \beta^\bullet\rangle \tag{2}$$

where Seq internalizes the meta-notation $\vdash_{\mathsf{IMLL}}$. We remark that (1) recalls:

$$[x]_o = x(\odot)\,.\,\overline{o}\langle\odot\rangle \tag{3}$$

the base clause of the, so called, *output-based embedding* of the *standard* λ-calculus *with explicit substitutions* into π-calculus [17]. In (3), "." is the sequential composition of the π-calculus, and $\circ$ a place-holder constant. In [8], (3) is called *forwarder*. So, SBV formulas contain the forwarder, the basic input/output communication flow that λ-variables realize, if we look at (1) as a process. However, we could not see how to obtain any *on-the-fly renaming* of channels able to model the substitution of a term for a bound variable, namely, β-reduction. So, we extended SBV to SBVr.

Contributions and paper organization.

System SBVr. The system SBVr (Section 2) extends SBV by adding Rename, a binary renaming operator $\lceil\cdot\rfloor_\cdot$. Rename is self-dual, it binds atoms, and it is the inverse of α-rule. The prominent defining axiom of Rename is $R \approx \lceil R\{^a/_b\}\rfloor_a$ (Figure 3) where R is a formula of SBVr. The meta-notation $R\{^a/_b\}$ denotes the capture-free substitution of a for b, and of $\overline{a}$ for $\overline{b}$ in R. Roughly, we can think of $\lceil R\{^a/_b\}\rfloor_a$ as $\forall a.R\{^a/_b\}$. The idea is that we shall exploit Rename for renaming atoms of formulas which represent linear λ-terms with explicit substitutions. Like a universal quantifier, Rename sets the boundary where the name change can take place without altering the set of free names of SBVr of a formula.

Completeness of SBVr *and* BVr. First we recall that linear λ-calculus embodies the core of functional programming. The functions it represents use their arguments exactly once in the course of the evaluation. The set of functions we can express in it are quite limited, but "large" enough to let the decision about which is the normal form of its λ-terms a *polynomial time complete* problem [10], if we take the polynomial time Turing machines as computational model of reference. We shall focus on λ-calculus *with explicit substitutions* [1]. "Explicit" means that the operation that substitutes a λ-term for a λ-variable, in the course of a β-reduction, is not meta, but a term constructor.

Concerning the completeness of SBVr, we start defining an embedding $(\!|\cdot|\!)_\cdot$ from λ-terms with explicit substitutions to formulas of SBVr (Figure 8, Section 5.) Then, we prove that, for every linear λ-term with explicit substitutions M, and every atom o, which plays the role of an output-channel, if M reduces to N, then there is a *derivation* of SBVr, with $(\!|M|\!)_o$ as conclusion, and $(\!|N|\!)_o$ as premise (Theorem 5.6.) Namely, completeness says that the evaluation of a linear λ-term with explicit substitutions M, viewed as a formula, is proof-search in SBVr. As a corollary, we show that BVr, the cut free subsystem of SBVr, is complete with respect to linear λ-calculus with explicit substitutions as well. We mean that a computation from M to N in BVr becomes a process of annihilation between the formula $(\!|M|\!)_o$ that represents M, and the *negation* of $(\!|N|\!)_o$, representing N (Corollary 5.7.)

It is worth remarking that the above embedding $(\!|\cdot|\!)_\cdot$ strongly recalls *output-based* embedding of *standard* λ-calculus into π-calculus [17]. The reason why we think this is relevant is simple. We recall that the traditional embeddings of *standard* λ-calculus into π-calculus follow, instead, the *input-based* pattern of [12]. Input-based embeddings are quite restrictive. The only β-reduction strategy they allow to simulate by π-calculus processes is the *lazy* one. Instead, output-based embedding extends the simulation to *spine* evaluation strategy, which strictly includes the lazy one. So, extending SBVr with logical operators that model replication of λ-terms will possibly extend completeness we now have for SBVr, relatively to linear λ-calculus with explicit substitutions, to wider subsets of full λ-calculus.

Cut elimination for SBVr. The above mentioned completeness of BVr follows from proving that the cut elimination holds inside SBVr (Theorem 3.6.) The main steps to prove cut elimination are *shallow splitting*, *context reduction*, *splitting*, and *admissibility of the up fragment*, following [6]. The cut elimination for SBVr says that its theorems can be proved by means of the rules of its subsystem BVr free of cut rule.

Soundness of SBVr. It says that proof-search inside SBVr is an interpreter of linear λ-terms with explicit substitutions. We show that, if a derivation of SBVr proves that $(\!|M|\!)_o$ derives from $(\!|N|\!)_o$ in SBVr, then M reduces N (Theorem 5.6.) However, the derivation of $(\!|M|\!)_o$ from $(\!|N|\!)_o$ must be under a specific proof-search strategy. This, might limit efficiency. We mean that all the freedom we might gain thanks to the deep application of the logical rules, in the course of a proof-search, can be lost by sticking to the specific strategy we are referring to and that we shall see.

Expressiveness of SBVr *and* BVr. Linear λ-calculus can compute all boolean functions [10]. Proof-search of SBVr, and BVr can do the same thanks to the above completeness. So, the extension of SBV to SBV`r` is not trivial. Moreover, since BV is NPtime-complete [9], so is BV`r`.

Extended version. This work is an extended abstract and contains a single full detailed proof. The technical report [13] contains all proofs of the coming statements, and a richer bibliography.

$$R ::= a \mid \overline{a} \mid \circ \mid [R \parr R] \mid (R \otimes R) \mid \langle R \triangleleft R \rangle \mid \ulcorner R \lrcorner_a$$

Fig. 1. Structures

2 Systems SBV`r` and BV`r`

Structures. Let $a, b, c, \ldots$ denote the elements of a countable set of *positive propositional variables*. Instead, $\overline{a}, \overline{b}, \overline{c}, \ldots$ denote the elements of a countable set of *negative propositional variables*. The set of *atoms* contains both positive and negative propositional variables, and nothing else. Let $\circ$ be a *constant* different from any atom. The grammar in Figure 1 gives the set of *structures*, the standard name for formulas in SBV. We shall range over structures with R, P, T, U, and V. Par $[R \parr R]$, CoPar $(R \otimes R)$, and Seq $\langle R \triangleleft R \rangle$ come from SBV. Rename $\ulcorner R \lrcorner_a$ is new and comes with the proviso that a must be a positive atom. Namely, $\ulcorner R \lrcorner_{\overline{a}}$ is not in the syntax. Rename implies the defini-

$$\begin{aligned}
&\{a\} = \text{FN}(a) \cup \text{FN}(\overline{a})\\
&a \in \text{FN}(\langle R \triangleleft T \rangle) \text{ if } a \in \text{FN}(R) \cup \text{FN}(T)\\
&a \in \text{FN}((R \otimes T)) \text{ if } a \in \text{FN}(R) \cup \text{FN}(T)\\
&a \in \text{FN}([R \parr T]) \text{ if } a \in \text{FN}(R) \cup \text{FN}(T)\\
&a \in \text{FN}(\ulcorner R \lrcorner_b) \text{ if } a \neq b \text{ and } a \in \text{FN}(R)
\end{aligned}$$

Fig. 2. Free names of structures.

tion of the *free names* $\text{FN}(R)$ of R as in Figure 2.

Size of the structures. The *size* $|R|$ of R sums the number of occurrences of atoms in R and the number of occurrences of renaming operators $\ulcorner T \lrcorner_a$ inside R whose bound variable a belongs to $\text{FN}(T)$. For example, $|\ulcorner[b \parr \overline{b}]\lrcorner_a| = 2$, while $|\ulcorner[a \parr \overline{a}]\lrcorner_a| = 3$.

Negation

$$\bar{\circ} \approx \circ$$
$$\overline{[R \parr T]} \approx (\bar{R} \otimes \bar{T})$$
$$\overline{(R \otimes T)} \approx [\bar{R} \parr \bar{T}]$$
$$\overline{\langle R \triangleleft T \rangle} \approx \langle \bar{R} \triangleleft \bar{T} \rangle$$
$$\overline{\lceil R \rfloor_a} \approx \lceil \bar{R} \rfloor_a$$

Renaming

$$R \approx \lceil R\{^a/_b\} \rfloor_a \text{ if } a \notin \mathrm{FN}(R)$$
$$\circ\{^a/_b\} \approx \circ$$
$$b\{^a/_b\} \approx a$$
$$\bar{b}\{^a/_b\} \approx \bar{a}$$
$$c\{^a/_b\} \approx c$$
$$\bar{c}\{^a/_b\} \approx \bar{c}$$
$$[R \parr T]\{^a/_b\} \approx [R\{^a/_b\} \parr T\{^a/_b\}]$$
$$(R \otimes T)\{^a/_b\} \approx (R\{^a/_b\} \otimes T\{^a/_b\})$$
$$\langle R \triangleleft T \rangle\{^a/_b\} \approx \langle R\{^a/_b\} \triangleleft T\{^a/_b\} \rangle$$
$$\lceil R \rfloor_b\{^a/_b\} \approx R$$
$$\lceil R \rfloor_c\{^a/_b\} \approx \lceil R\{^a/_b\} \rfloor_c$$

Contextual Closure

$$S\{R\} \approx S\{T\} \text{ if } R \approx T$$

Unit

$$R \approx [\circ \parr R] \approx (\circ \otimes R)$$
$$R \approx \langle \circ \triangleleft R \rangle \approx \langle R \triangleleft \circ \rangle$$

Associativity

$$[[R \parr T] \parr V] \approx [R \parr [T \parr V]]$$
$$((R \otimes T) \otimes V) \approx (R \otimes (T \otimes V))$$
$$\langle \langle R \triangleleft T \rangle \triangleleft V \rangle \approx \langle R \triangleleft \langle T \triangleleft V \rangle \rangle$$

Symmetry

$$[R \parr T] \approx [T \parr R]$$
$$(R \otimes T) \approx (T \otimes R)$$

Distributivity

$$\lceil \langle R \triangleleft U \rangle \rfloor_a \approx \langle \lceil R \rfloor_a \triangleleft \lceil U \rfloor_a \rangle$$

Exchange

$$\lceil \lceil R \rfloor_b \rfloor_a \approx \lceil \lceil R \rfloor_a \rfloor_b$$

Fig. 3. Equivalence $\approx$ on structures.

Equivalence on structures. Structures are equivalent up to the smallest congruence defined by the set of axioms in Figure 3 where Rename is a self-dual operator. By $R\{^a/_b\}$ we denote the capture-free substitution of a for b, and of $\overline{a}$ for $\overline{b}$ in R. The intuitive reason why Rename is self-dual follows. Since $R\{^a/_b\}$ denotes R where every free occurrence of the atom a, and its dual $\overline{a}$, replaces b, and $\overline{b}$, respectively, nothing changes when operating on $\overline{R}$ in place of R. Every occurrence of a in $\overline{R}$ corresponds to one of $\overline{a}$ of R, and vice-versa. We observe that $\circ \approx \lceil \circ \rfloor_a$ is as particular case of $R \approx \lceil R\{^a/_a\} \rfloor_a \approx \lceil R \rfloor_a$, which holds whenever $a \notin \mathrm{FN}(R)$. Finally, Distributivity in Figure 3 must be understood in relation to the deductive rules of $\mathsf{SBV}\mathtt{r}$, so we postpone the discussion.

(Structure) Contexts. They are $S\{\ \}$, i.e. a structure with a single hole $\{\ \}$ in it. If $S\{R\}$, then R is a *substructure* of S. For example, we shall tend to shorten $S\{[R \parr U]\}$ as $S[R \parr U]$ when $[R \parr U]$ fills the hole $\{\ \}$ of $S\{\ \}$ exactly.

$$\mathsf{ai}{\downarrow}\,\frac{\circ}{[a \parr \overline{a}]} \qquad\qquad \mathsf{ai}{\uparrow}\,\frac{(a \otimes \overline{a})}{\circ}$$

$$\mathsf{q}{\downarrow}\,\frac{\langle [R \parr U] \lhd [T \parr V]\rangle}{[\langle R \lhd T\rangle \parr \langle U \lhd V\rangle]} \qquad \mathsf{s}\,\frac{([R \parr T] \otimes U)}{[(R \otimes U) \parr T]} \qquad \mathsf{q}{\uparrow}\,\frac{(\langle R \lhd T\rangle \otimes \langle U \lhd V\rangle)}{\langle (R \otimes U) \lhd (T \otimes V)\rangle}$$

$$\mathsf{r}{\downarrow}\,\frac{\lceil [R \parr U] \rfloor_a}{[\lceil R \rfloor_a \parr \lceil U \rfloor_a]} \qquad\qquad \mathsf{r}{\uparrow}\,\frac{(\lceil R \rfloor_a \otimes \lceil U \rfloor_a)}{\lceil (R \otimes U) \rfloor_a}$$

Fig. 4. System $\mathsf{SBV}\mathtt{r}$.

The system $\mathsf{SBV}\mathtt{r}$. It contains the set of inference rules in Figure 4 with form $\rho\,\dfrac{T}{R}$, *name* ρ, *premise* T, and *conclusion* R. The typical use of an inference rules is $\rho\,\dfrac{S\{T\}}{S\{R\}}$. It specifies that if a structure U matches R in a context $S\{\ \}$, it can be rewritten to $S\{T\}$. Since rules apply in any context, and we use them as rewriting rules, R is the *redex* of ρ, and T its *reduct*.

$\mathsf{BV}\mathtt{r}$ is the *down fragment* $\{\mathsf{ai}{\downarrow}, \mathsf{s}, \mathsf{q}{\downarrow}, \mathsf{r}{\downarrow}\}$ of $\mathsf{SBV}\mathtt{r}$. The *up fragment* of $\mathsf{SBV}\mathtt{r}$ is $\{\mathsf{ai}{\uparrow}, \mathsf{s}, \mathsf{q}{\uparrow}, \mathsf{r}{\uparrow}\}$. So s belongs to both. The rules that formalize Rename are $\mathsf{r}{\downarrow}$, and $\mathsf{r}{\uparrow}$. The former can be viewed as the restriction to a self-dual quantifier of the rule $\mathsf{u}{\downarrow}$ which, in [14], formalizes the standard universal quantifier.

Derivation and proof. A *derivation* in $\mathsf{SBV}\mathtt{r}$ is either a structure or an instance of the above rules or a sequence of two derivations. The topmost structure in a derivation is its *premise*. The bottommost is its *conclusion*. The *length* $|\mathscr{D}|$ of a derivation $\mathscr{D}$ is the number of rule instances in $\mathscr{D}$. A derivation $\mathscr{D}$ of a structure R in $\mathsf{SBV}\mathtt{r}$ from a structure

T in $\mathsf{SBV}\mathtt{r}$, only using a subset $\mathsf{B} \subseteq \mathsf{SBV}\mathtt{r}$ is $\begin{array}{c} T \\ \mathscr{D}\|\mathsf{B} \\ R \end{array}$. The equivalent *space-saving* form we shall tend to use is $\mathscr{D} : T \vdash_{\mathsf{B}} R$. The derivation $\begin{array}{c} T \\ \mathscr{P}\|\mathsf{B} \\ R \end{array}$ is a *proof* whenever $T \approx \circ$. We denote it as $\begin{array}{c} \mathscr{P}\|\mathsf{B} \\ R \end{array}$, or $\begin{array}{c} \circ \\ \mathscr{P}\|\mathsf{B} \\ R \end{array}$, or $\mathscr{P} : \vdash_{\mathsf{B}} R$. In general, we shall drop B when clear from the context. We shall write $\dfrac{T}{R}$ (with $=$) to mean $R \approx T$.

We are in the position to give an account of Distributivity in Figure 3. It morally embodies two rules, one flipping the other:

$$\frac{\ulcorner\langle R \triangleleft U\rangle\lrcorner_a}{\langle\ulcorner R\lrcorner_a \triangleleft \ulcorner U\lrcorner_a\rangle} \qquad \frac{\langle\ulcorner R\lrcorner_a \triangleleft \ulcorner U\lrcorner_a\rangle}{\ulcorner\langle R \triangleleft U\rangle\lrcorner_a}$$

Replacing these two rules by a single equivalence axiom is coherent to the intuitive interpretation of every Seq structure $\langle R \triangleleft U\rangle$ in [6]. Whatever the derivation $\mathscr{D}$ that contains $\langle R \triangleleft U\rangle$ is, neither going upward in $\mathscr{D}$, nor moving downward in it, any atom of R will ever interact with any atom of U. So, Rename can indifferently operate locally to R, and U or globally on $\langle R \triangleleft U\rangle$.

The following proposition shows when two structures R, T can be "moved" inside a context so that they are one aside the other and may eventually communicate going upward in a derivation.

Proposition 2.1 (***Context extrusion.***) *$S[R \parr T] \vdash_{\{\mathsf{q}\downarrow,\mathsf{s},\mathsf{r}\downarrow\}} [S\{R\} \parr T]$, for every S, R, T.*

Equivalence of systems. A subset $\mathsf{B} \subseteq \mathsf{SBV}\mathtt{r}$ *proves* R if $\mathscr{P} : \vdash_{\mathsf{B}} R$, for some proof $\mathscr{P}$. Two subsets B and B' of $\mathsf{SBV}\mathtt{r}$ are *strongly equivalent* if, for every derivation $\mathscr{D} : T \vdash_{\mathsf{B}} R$, there exists a derivation $\mathscr{D}' : T \vdash_{\mathsf{B}'} R$, and vice versa. Two *systems are equivalent* if they prove the same structures.

Derivable rules. A rule ρ is *derivable* in $\mathsf{B} \subseteq \mathsf{SBV}\mathtt{r}$ if $\rho \notin \mathsf{B}$ and, for every instance $\rho\,\dfrac{T}{R}$, there exists a derivation $\mathscr{D}$ in B such that $\mathscr{D} : T \vdash_{\mathsf{B}} R$. Figure 5 recalls a core set

$$\mathsf{i}\!\downarrow\frac{\circ}{[R \parr \overline{R}]} \qquad \mathsf{i}\!\uparrow\frac{(R \otimes \overline{R})}{\circ} \qquad \mathsf{mixp}\,\frac{(R \otimes T)}{\langle R \triangleleft T\rangle}$$

$$\mathsf{def}\!\downarrow\frac{\langle R \triangleleft T\rangle}{[\langle R \triangleleft \overline{a}\rangle \parr (a \otimes T)]} \qquad \mathsf{def}\!\uparrow\frac{(\langle R \triangleleft \overline{a}\rangle \otimes [a \parr T])}{\langle R \triangleleft T\rangle} \qquad \mathsf{pmix}\,\frac{\langle R \triangleleft T\rangle}{[R \parr T]}$$

Fig. 5. A core-set of rules derivable in $\mathsf{SBV}\mathtt{r}$.

of rules derivable in SBV, hence in SBVr. *General interaction down* and *up* rules are $\mathsf{i}\downarrow$, and $\mathsf{i}\uparrow$, respectively. The rule $\mathsf{def}\downarrow$ uses $\overline{a}$ as a place-holder, and a as name for T. Building the derivation upward, we literally replace T for $\overline{a}$. Symmetrically for $\mathsf{def}\uparrow$. The rules mixp, and pmix show a hierarchy between the connectives, where $\parr$ is the lowermost, $\triangleleft$ lies in the middle, and $\otimes$ on top. General interaction up is derivable in $\{\mathsf{ai}\uparrow, \mathsf{s}, \mathsf{q}\uparrow, \mathsf{r}\uparrow\}$, while $\mathsf{def}\downarrow$ is derivable in $\{\mathsf{ai}\downarrow, \mathsf{s}, \mathsf{q}\downarrow\}$, and mixp is derivable in $\{\mathsf{q}\uparrow\}$:

$$\begin{array}{rc} & (\lceil R\rfloor_a \otimes \overline{\lceil R\rfloor_a}) \\ \hline\hline & (\lceil R\rfloor_a \otimes \lceil \overline{R}\rfloor_a) \\ \hline \mathsf{r}\uparrow & \lceil (R \otimes \overline{R})\rfloor_a \\ \hline \mathsf{i}\uparrow & \lceil \circ \rfloor_a \quad \text{ind. hypothesis} \\ \hline\hline & \circ \end{array} \qquad \begin{array}{rc} & \langle R \triangleleft T\rangle \\ \hline\hline & \langle R \triangleleft (\circ \otimes T)\rangle \\ \hline \mathsf{ai}\downarrow & \langle R \triangleleft ([a \parr \overline{a}] \otimes T)\rangle \\ \hline\hline \mathsf{s} & \langle [R \parr \circ] \triangleleft [a \parr (\overline{a} \otimes T)]\rangle \\ \hline \mathsf{q}\downarrow & [\langle R \triangleleft a\rangle \parr \langle \circ \triangleleft (\overline{a} \otimes T)\rangle] \\ \hline\hline & [\langle R \triangleleft a\rangle \parr (\overline{a} \otimes T)] \end{array} \qquad \begin{array}{rc} & (R \otimes T) \\ \hline\hline & (\langle R \triangleleft \circ\rangle \otimes \langle \circ \triangleleft T\rangle) \\ \hline \mathsf{q}\uparrow & \langle (R \otimes \circ) \triangleleft (\circ \otimes T)\rangle \\ \hline\hline & \langle R \triangleleft T\rangle \end{array}$$

The leftmost derivation sketches how to derive general interaction up in the case new to SBVr, as compared to SBV. Symmetrically, *general interaction down* is derivable in $\{\mathsf{ai}\downarrow, \mathsf{s}, \mathsf{q}\downarrow, \mathsf{r}\downarrow\}$, while $\mathsf{def}\uparrow$ in $\{\mathsf{ai}\uparrow, \mathsf{s}, \mathsf{q}\uparrow\}$, and pmix in $\{\mathsf{q}\downarrow\}$.

3 Cut elimination of SBVr

The goal is to prove that SBVr, and BVr are equivalent. Proving the equivalence, amounts to proving that every up rule is admissible in BVr or, equivalently, that we can eliminate them from any derivation of SBVr. Splitting theorem for BVr, which extends the namesake theorem for BV [6], is the effective tool we prove to exist for showing that the up fragment of SBVr is admissible for BVr.

Proposition 3.1 (BVr ***is affine.***) *In every $\mathscr{D} : T \vdash R$, we have $|R| \geq |T|$.*

Proposition 3.2 here below says that the components of Seq, and CoPar can be proved independently, and that Rename is a universal quantifier on atoms.

Proposition 3.2 (***Derivability of structures in*** **BVr.**) *For all R, T, and a:*

1. *$\mathscr{P} : \vdash \langle R \triangleleft T\rangle$ iff $\mathscr{P}_1 : \vdash R$ and $\mathscr{P}_2 : \vdash T$.*
2. *$\mathscr{P} : \vdash (R \otimes T)$ iff $\mathscr{P}_1 : \vdash R$ and $\mathscr{P}_2 : \vdash T$.*
3. *$\mathscr{P} : \vdash \lceil R\rfloor_a$ iff $\mathscr{P}' : \vdash R\{^b/_a\}$, for every atom b.*

Proposition 3.3 here below says that part of the structure in the conclusion of every proof of BVr, which in the proposition will be P, is the context that, suitably decomposed, allows to annihilate each component of the remaining part of the conclusion.

Proposition 3.3 (***Shallow Splitting in*** **BVr.**) *For all R, T, P, and a:*

1. *If $\mathscr{P} : \vdash [\langle R \triangleleft T\rangle \parr P]$, then $\langle P_1 \triangleleft P_2\rangle \vdash P$, and $\vdash [R \parr P_1]$, and $\vdash [T \parr P_2]$, for some P_1, P_2.*
2. *If $\mathscr{P} : \vdash [(R \otimes T) \parr P]$, then $[P_1 \parr P_2] \vdash P$, and $\vdash [R \parr P_1]$, and $\vdash [T \parr P_2]$, for some P_1, P_2.*
3. *If $\mathscr{P} : \vdash [a \parr P]$, then $\mathscr{P}' : \overline{a} \vdash P$.*

4. *If $\mathscr{P} : \vdash [\lceil R \rfloor_a \parr P]$, then $\lceil P' \rfloor_a \vdash P$, and $\vdash [R \parr P']$, for some P'.*

Proposition 3.4 here below says that in the conclusion of every proof of BVr we can always focus on a specific structure R so that, upward in the proof, a suitable structure U allows to annihilate R. If some Rename gets in the way in between R, and U, we can always move it outward so that it does not interfere.

Proposition 3.4 (***Context Reduction in*** **BVr.**) *For all R and contexts $S\{\ \}$ such that $\mathscr{P} : \vdash S\{R\}$, there are U, a such that $\mathscr{D} : \lceil [\{\ \} \parr U] \rfloor_a \vdash S\{\ \}$, and $\vdash [R \parr U]$.*

Namely, here above, $S\{\ \}$ supplies the "context" U, required for annihilating R, no matter which structure fills the hole of $S\{\ \}$, and no matter the existence of Rename enclosing R and U.

Theorem 3.5 here below results from composing Context Reduction (Proposition 3.4), and Shallow Splitting (Proposition 3.3) in this order. It says that in every proof of BVr we can always focus on any structure inside the conclusion and, moving upward in the proof, we can annihilate it, the final goal being obtaining only $\circ$ as premise of the whole proof.

Theorem 3.5 (***Splitting in*** **BVr.**) *For all R, T, and contexts $S\{\ \}$:*

1. *If $\mathscr{P} : \vdash S\langle R \lhd T \rangle$, then $\lceil [\{\ \} \parr \langle K_1 \lhd K_2 \rangle] \rfloor_a \vdash S\{\ \}$, and $\vdash [R \parr K_1]$, and $\vdash [T \parr K_2]$, for some K_1, K_2, a.*
2. *If $\mathscr{P} : \vdash S(R \otimes T)$, then $\lceil [\{\ \} \parr [K_1 \parr K_2]] \rfloor_a \vdash S\{\ \}$, and $\vdash [R \parr K_1]$, and $\vdash [T \parr K_2]$, for some K_1, K_2, a.*
3. *If $\mathscr{P} : \vdash S\lceil R \rfloor_a$, then $\lceil [\{\ \} \parr K] \rfloor_a \vdash S\{\ \}$, and $\vdash [R \parr K]$, for some K, a.*

Theorem 3.6 here below results from composing Splitting (Theorem 3.5), and Shallow Splitting (Proposition 3.3). Just to give an informal idea about its meaning, in its proof we can show that every derivation of SBVr that contains an instance of $\mathsf{r}\uparrow$ can be rewritten as a derivation with the same conclusion, without the above occurrence of $\mathsf{r}\uparrow$, but with a couple of new instances of both $\mathsf{r}\downarrow$, and s.

Theorem 3.6 (***Admissibility of the up fragment for*** **BVr.**) *The set of rules $\{\mathsf{ai}\uparrow, \mathsf{q}\uparrow, \mathsf{r}\uparrow\}$ of SBVr is admissible for BVr.*

Theorem 3.6 here above directly implies:

Corollary 3.7 *The cut elimination holds for SBVr.*

4 Linear λ-calculus with explicit substitutions

Linear λ-calculus with explicit substitutions is a pair with a set of linear λ-terms, and an operational semantics on them. The operational semantics looks at substitution as explicit syntactic component and not as meta-operation.

The linear λ-terms. Let $\mathscr{V}$ be a countable set of variable names we range over by x, y, w, z. We call $\mathscr{V}$ the *set of λ-variables*. The set of *linear λ-terms with explicit substitutions* is $\Lambda = \bigcup_{X \subset \mathscr{V}} \Lambda_X$ we range over by M, N, P, Q. For every $X \subset \mathscr{V}$, the set Λ_X contains the *linear λ-terms with explicit substitutions whose free variables are in* X, and which we define as follows: (i) $x \in \Lambda_{\{x\}}$; (ii) $\lambda x.M \in \Lambda_X$ if $M \in \Lambda_{X \cup \{x\}}$; (iii) $(M)\,N \in \Lambda_{X \cup Y}$ if $M \in \Lambda_X$, $N \in \Lambda_Y$, and $X \cap Y = \emptyset$; (iv) $(M)\,\{x = P\} \in \Lambda_{X \cup Y}$ if $M \in \Lambda_{X \cup \{x\}}$, $P \in \Lambda_Y$, and $X \cap Y = \emptyset$.

$$
\begin{array}{rll}
(\lambda x.M)\,N & \to (M)\,\{x = N\} & \\
(x)\,\{x = P\} & \to P & \\
(\lambda y.M)\,\{x = P\} & \to \lambda y.(M)\,\{x = P\} & \\
((M)\,N)\,\{x = P\} & \to (M)\,(N)\,\{x = P\} & \text{if } x \in \mathrm{fv}(N) \\
((M)\,N)\,\{x = P\} & \to ((M)\,\{x = P\})\,N & \text{if } x \in \mathrm{fv}(M)
\end{array}
$$

Fig. 6. β-reduction $\to \subseteq \Lambda \times \Lambda$ with explicit substitution.

β-reduction on linear λ-terms with explicit substitutions. It is the relation $\to$ in Figure 6. It is the core of the very simple, indeed, computations the syntax of the terms in Λ allow to develop. The point, however, is that the core computational mechanism that replaces a term for a variable is there, and we aim at modeling it inside SBVr. Moreover, despite Λ is so simple, it can model all the boolean functions [10].

$$
\mathsf{rfl}\,\frac{}{M \Rightarrow M}
\qquad
\mathsf{lft}\,\frac{M \to N}{M \Rightarrow N}
\qquad
\mathsf{tra}\,\frac{M \Rightarrow P \qquad P \Rightarrow N}{M \Rightarrow N}
$$

$$
\mathsf{f}\,\frac{M \Rightarrow N}{\lambda x.M \Rightarrow \lambda x.N}
\qquad
@\mathsf{l}\,\frac{M \Rightarrow N}{(M)\,P \Rightarrow (N)\,P}
\qquad
@\mathsf{r}\,\frac{M \Rightarrow N}{(P)\,M \Rightarrow (P)\,N}
$$

$$
\sigma\mathsf{l}\,\frac{M \Rightarrow N}{(M)\,\{x = P\} \Rightarrow (N)\,\{x = P\}}
\qquad
\sigma\mathsf{r}\,\frac{M \Rightarrow N}{(P)\,\{x = M\} \Rightarrow (P)\,\{x = N\}}
$$

Fig. 7. Rewriting relation $\Rightarrow \subseteq \Lambda \times \Lambda$.

Operational semantics on linear λ-terms with explicit substitutions. It is the relation $\Rightarrow$ in Figure 7, i.e. the reflexive, contextual, and transitive closure of the above β-reduction with explicit substitution. The number of instances of rules in Figure 7 that yield $M \Rightarrow N$ is $|M \Rightarrow N|$.

5 Completeness and Soundness of SBVr and BVr

We relate functional and proof-theoretic worlds. First we map terms of Λ into structures of SBVr. Then, we show the *completeness* of SBVr and BVr, i.e. that the computations of Λ correspond to proof-search inside the two systems. Finally, we prove *soundness* of SBVr with respect to the computations of λ-calculus with explicit substitutions under a specific proof-search strategy. This means that we can use SBVr to compute any term which any given M reduces to.

The map $\llparenthesis \cdot \rrparenthesis_{\cdot}$. We start with the following "fake" map from Λ to SBVr:

$$\llparenthesis x \rrparenthesis_o = \langle x \triangleleft \overline{o} \rangle \tag{4}$$
$$\llparenthesis \lambda x.M \rrparenthesis_o = \forall x.\exists p.[\llparenthesis M \rrparenthesis_p \parr (p \otimes \overline{o})] \tag{5}$$
$$\llparenthesis (M)\, N \rrparenthesis_o = \exists p.[\llparenthesis M \rrparenthesis_p \parr \exists q.\llparenthesis N \rrparenthesis_q \parr (p \otimes \overline{o})] \tag{6}$$
$$\llparenthesis (M)\,\{x = P\} \rrparenthesis_o = \forall x.[\llparenthesis M \rrparenthesis_o \parr \llparenthesis P \rrparenthesis_x] \tag{7}$$

We use it only to intuitively illustrate how we shall effectively represent terms of Λ as structures of SBVr. The map here above translates M into $\llparenthesis M \rrparenthesis_o$ where o is a unique output channel, while the whole expression depends on a set of free input channels, each for every free variable of M. Clause (4) associates the input channel x to the fresh output channel $\overline{o}$, under the intuition that x is *forwarded* to o, using the terminology of [8]. Clause (5) assumes $\llparenthesis M \rrparenthesis_p$ has p as output and (at least) x as input. It renames p, hidden by $\exists$, as $\overline{o}$ thanks to $(p \otimes \overline{o})$. This must work for every input x. For this reason we hide x by means of $\forall$. Clause (6) makes the output channels of both $\llparenthesis M \rrparenthesis_p$ and $\llparenthesis N \rrparenthesis_q$ local, while renaming p to $\overline{o}$ thanks to $(p \otimes \overline{o})$. If $\llparenthesis M \rrparenthesis_p$ will result in the translation of a λ-abstraction $\lambda z.P$, then the existential quantifier immediately preceding $\llparenthesis N \rrparenthesis_q$ will interact with the universal quantifier in front of $\llparenthesis M \rrparenthesis_p$. The result will be an on-the-fly channel name renaming. Clause (7) identifies the output of $\llparenthesis P \rrparenthesis_x$ with one of the existing free names of $\llparenthesis M \rrparenthesis_o$. The identification becomes local thanks to the universal quantifier.

$$\begin{aligned}
\llparenthesis x \rrparenthesis_o &= \langle x \triangleleft \overline{o} \rangle \\
\llparenthesis \lambda x.M \rrparenthesis_o &= \lceil \lceil [\llparenthesis M \rrparenthesis_p \parr (p \otimes \overline{o})] \rfloor_p \rfloor_x \\
\llparenthesis (M)\, N \rrparenthesis_o &= \lceil [\llparenthesis M \rrparenthesis_p \parr \lceil \llparenthesis N \rrparenthesis_q \rfloor_q \parr (p \otimes \overline{o})] \rfloor_p \\
\llparenthesis (M)\,\{x = P\} \rrparenthesis_o &= \lceil [\llparenthesis M \rrparenthesis_o \parr \llparenthesis P \rrparenthesis_x] \rfloor_x
\end{aligned}$$

Fig. 8. Map $\llparenthesis \cdot \rrparenthesis$ from Λ to structures

In a setting where the second order quantifiers $\forall$, and $\exists$ only operate on atoms, distinguishing between the two is meaningless. So, the renaming can be self-dual and the true map $\llparenthesis \cdot \rrparenthesis_{\cdot}$ which adheres to the above intuition is in Figure 8.

Remark 5.1 We keep stressing that $\llparenthesis \cdot \rrparenthesis_{\cdot}$ strongly recalls output-based embedding of *standard* λ-calculus with explicit substitutions into π-calculus [17]. In principle, this means that extending SBVr with the right logical operators able to duplicate atoms, and consequently upgrading $\llparenthesis \cdot \rrparenthesis_{\cdot}$, we could model full β-reduction as proof-search.

Lemma 5.2 **(*Output names are linear.*)** *Every output name of* $\llparenthesis M \rrparenthesis_o$ *occurs once in it.*

Lemma 5.3 **(*Output renaming.*)** *For every* $M, o,$ *and* p *we can derive* o−ren *in* BVr.

$$\text{o-ren}\,\frac{(\!|M|\!)_o}{[(\!|M|\!)_p \mathbin{\bindnasrepma} (p \otimes \overline{o})]} \qquad \text{s-var}\,\frac{(\!|P|\!)_o}{(\!|(x)\,\{x = P\}|\!)_o}$$

$$\text{s-intro}\,\frac{(\!|(M)\,\{x = N\}|\!)_o}{(\!|(\lambda x.M)\,N|\!)_o} \qquad \text{s-}\lambda\,\frac{(\!|\lambda y.(M)\,\{x = P\}|\!)_o}{(\!|(\lambda y.M)\,\{x = P\}|\!)_o}$$

$$\text{s-@l}\,\frac{(\!|((M)\,\{x = P\})\,N|\!)_o \quad x \in \mathrm{fv}(M)}{(\!|((M)\,N)\,\{x = P\}|\!)_o} \qquad \text{s-@r}\,\frac{(\!|(M)\,(N)\,\{x = P\}|\!)_o \quad x \in \mathrm{fv}(N)}{(\!|((M)\,N)\,\{x = P\}|\!)_o}$$

Fig. 9. Derivable rules that simulate β-reduction with explicit substitutions

Lemma 5.4 (***Simulating*** $\rightarrow$.) *For every M, N, P, o, p, and q, we can derive:*

1. s−intro, s−λ, s−@l, *and* s−@r *in* BVr, *and*
2. s−var *in* BVr $\cup$ {q↑}.

Appendix A shows how proving Lemma 5.4 here above, by detailing out the derivations of SBVr that derive the rules in Figure 9.

Remark 5.5 Were the clause "$(y)\,\{x = P\} \rightarrow y$" in the definition of $\rightarrow$ we could not prove Lemma 5.4 because $(\!|P|\!)_o \vdash (\!|(y)\,\{x = P\}|\!)_o$ would not exist in SBVr. The reason is that, given $(\!|(y)\,\{x = P\}|\!)_o$, it is not evident which logical tool can erase any translation of P as directly as happens in $(y)\,\{x = P\} \rightarrow y$. The only erasure mechanism existing in BVr is atom annihilation through the rules ai↓, and ai↑, indeed.

Theorem 5.6 here below says that to every β-reduction sequence from M to N in linear λ-calculus with explicit substitutions corresponds at least a proof-search process inside SBVr that builds a derivation with $(\!|N|\!)_o$ as premise and $(\!|M|\!)_o$ as conclusion.

Theorem 5.6 (***Completeness of*** SBVr.) *For every M, and o, if $M \Rightarrow N$, then there is $\mathscr{D} : (\!|N|\!)_o \vdash (\!|M|\!)_o$ in* SBVr, *where* q ↑ *is the only rule of the up-fragment in* SBVr *that can occur in $\mathscr{D}$.*

We remark that the strategy to derive $(\!|M|\!)_o$ from $(\!|N|\!)_o$ in SBVr has no connection to the one we might use for rewriting M to N. Corollary 5.7 here below follows from Theorem 5.6 and cut elimination of SBVr (Corollary 3.7). It reinterprets logically the meaning of rewriting M to N. It says that M transforms to N if $\overline{(\!|N|\!)_o}$ logically annihilate the components of $(\!|M|\!)_o$.

Corollary 5.7 (***Completeness of*** BVr.) *For every M, N, and o, if $M \Rightarrow N$, then, in* BVr, *we have* $\vdash [(\!|M|\!)_o \mathbin{\bindnasrepma} \overline{(\!|N|\!)_o}]$.

Theorem 5.8 here below says that proof-search inside SBVr can be used as an interpreter of linear λ-calculus with explicit substitutions. We say it is *Weak* Soundness because the interpreter follows a specific strategy of proof-search. The strategy is made of somewhat rigid steps represented by the rules in Figure 9 that simulate β-reduction.

Theorem 5.8 **(*Weak Soundness of* SBVr.)** *For every M, N, and o, let $\mathscr{D} : (\!| N |\!)_o \vdash (\!| M |\!)_o$ in* SBVr *be derived by composing a, possibly empty, sequence of rules in Figure 9. Then $M \Rightarrow N$.*

Conjecture 5.9 **(*Soundness of* SBVr *and* BVr.)** *As a referee suggested, Theorem 5.8 above should hold by dropping the requirement that, for building $\mathscr{D}$ in* SBVr*, we forcefully have to compose rules in Figure 9. The proof of this conjecture could take advantage of the method that works for proving that* BV*, extended with exponentials of linear logic, is undecidable [16].*

A corollary of the conjecture would say that, for every M, N, and o, if $\mathscr{P} : \vdash [(\!| M |\!)_o \parr \overline{(\!| N |\!)_o}]$ in BVr*, then $M \Rightarrow N$. Namely, we could use the process that searches the shortest cut free proof of $[(\!| M |\!)_o \parr \overline{(\!| N |\!)_o}]$ in* BVr *as an interpreter of linear λ-calculus with explicit substitutions. Of course, the possible relevance of this would be evident in a further extension of* BVr *where* full *β-reduction could be simulated as proof-search of cut free proofs.*

6 Conclusions and future work

We define an extension SBVr of SBV by introducing an atom renaming operator Rename which is a self-dual limited version of universal and existential quantifiers. Rename and Seq model the evaluation of linear λ-terms with explicit substitutions as proof-search in SBVr. So, we do not apply DI methodology to reformulate an existing logical system we already know to enjoy Curry-Howard correspondence with λ-calculus. Instead, we use logical operators at the core of DI, slightly extended, to get a computational behavior we could not obtain otherwise.

We conclude with a possible list of natural developments.

Of course proving Conjecture 5.9 is one of the obvious goals. Proving it would justify the relevance of looking for further extensions of SBVr whose unconstrained proof-search strategies could supply sound interpreters of *full* λ-calculus (with explicit substitutions). Starting points could be [16, 7, 15].

Also, we can think of extending SBVr by an operator that models non-deterministic choice. One reason would be proving the following statement. Let us assume we know that a λ-term M can only reduce to one of the normal forms $N_1, \ldots, N_m$. Let us assume the following statement can hold in the hypothetical extension of SBVr:

If $[\overline{(\!| P_1 |\!)_o} \oplus \cdots \oplus \overline{(\!| P_{i-1} |\!)_o} \oplus \overline{(\!| P_{i+1} |\!)_o} \oplus \cdots \oplus \overline{(\!| P_m |\!)_o}] \vdash [(\!| M |\!)_o \parr [\overline{(\!| N_1 |\!)_o} \oplus \cdots \oplus \overline{(\!| N_m |\!)_o}]]$,
then M reduces to N_i, for some $P_1, \ldots, P_m$.

This way we would represent the evaluation space of any linear λ-term with explicit substitutions as a non-deterministic process searching for normal forms. Candidate rules for non-deterministic choice to extend SBVr could be[1]:

$$\mathsf{p}{\downarrow}\,\frac{[[R \parr T] \oplus [U \parr T]]}{[[R \oplus U] \parr T]} \qquad\qquad \mathsf{p}{\uparrow}\,\frac{([R \oplus U] \otimes T)}{[(R \otimes T) \oplus (U \otimes T)]}$$

[1] The conjecture about the existence of the two rules p ↓, and p ↑, that model non-deterministic choice, results from discussions with Alessio Guglielmi.

A further reason to extend SBVr with non-deterministic choice is to keep developing the programme started in [4], aiming at a purely logical characterization of *full* CCS.

Finally, we could explore if any relation between linear λ-calculus with explicit substitutions, as we embed it in SBVr using a calculus-of-process style, and the evolution of quantum systems as proofs of BV [2], exists. Exploration makes sense if we observe that modeling a λ-variable x as a forwarder $\langle x \triangleleft \overline{o} \rangle$ is, essentially, looking at x as a sub-case of $\langle (x_1 \otimes \cdots \otimes x_k) \triangleleft [\overline{o}_1 \parr \cdots \parr \overline{o}_l] \rangle$, the representation of edges in DAGs that model quantum systems evolution in [2].

References

1. M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit substitutions. *JFP*, 1(4):375–416, 1991.
2. R. F. Blute, A. Guglielmi, I. T. Ivanov, P. Panangaden, and L. Straßburger. A Logical Basis for Quantum Evolution and Entanglement. Private communication.
3. K Brünnler and R. McKinley. An algorithmic interpretation of a deep inference system. In I. Cervesato, H. Veith, and A. Voronkov, editors, *LPAR08*, volume 5330 of *LNCS*, pages 482–496. Springer, 2008.
4. P. Bruscoli. A purely logical account of sequentiality in proof search. In P. J. Stuckey, editor, *Logic Programming, 18th International Conference*, volume 2401 of *LNCS*, pages 302–316. Springer, 2002.
5. J.-Y. Girard, P. Taylor, and Y. Lafont. *Proofs and Types*. CUP, 1989.
6. A. Guglielmi. A system of interaction and structure. *ToCL*, 8(1):1–64, 2007.
7. A. Guglielmi and L. Straßburger. A system of interaction and structure V: The exponentials and splitting. To appear on MSCS, 2009.
8. K. Honda and N. Yoshida. On the Reduction-based Process Semantics. *TCS*, (151):437–486, 1995.
9. O. Kahramanoğulları. System BV is NP-complete. *APAL*, 152(1–3):107–121, 2007.
10. H.G. Mairson. Linear lambda calculus and ptime-completeness. *JFP*, 14(6):623–633, 2004.
11. R. Milner. *Communication and Concurrency*. International Series in Computer Science. Prentice Hall, 1989.
12. R. Milner. Functions as processes. *MSCS*, 2(2):119–141, 1992.
13. L. Roversi. Linear lambda calculus with explicit substitutions as proof-search in Deep Inference. http://arxiv.org/abs/1011.3668. November 2010.
14. L. Straßburger. Some observations on the proof theory of second order propositional multiplicative linear logic. In P.-L. Curien, editor, *TLCA09*, volume 5608 of *LNCS*, pages 309–324. Springer, 2009.
15. L. Straßburger and A. Guglielmi. A system of interaction and structure IV: The exponentials and decomposition. *ToCL*, 2010. In press.
16. Lutz Straßburger. System NEL is undecidable. In Ruy De Queiroz, Elaine Pimentel, and Lucília Figueiredo, editors, *WoLLIC2003*, volume 84 of *ENTCS*. Elsevier, 2003.
17. S. van Bakel and M.G. Vigliotti. A logical interpretation of the λ-calculus into the π-calculus, preserving spine reduction and types. In *CONCUR09*, volume 5710 of *LNCS*, pages 84–98. Springer, 2009.

A Proof of Lemma 5.4

We simulate s−var with the following derivation:

$$
\begin{array}{rc}
 & \llparenthesis P \rrparenthesis_o \\
 & \mathscr{D}\Big\Vert \\
 & \lceil[\llparenthesis P \rrparenthesis_x \parr (x \otimes \overline{o})]\rfloor_x \\ \cline{2-2}
\mathsf{mixp} & \lceil[\llparenthesis P \rrparenthesis_x \parr \langle x \triangleleft \overline{o}\rangle]\rfloor_x \\ \hline\hline
\mathsf{e}_0 & \llparenthesis (x)\,\{x = P\} \rrparenthesis_o \equiv \lceil[\llparenthesis x \rrparenthesis_o \parr \llparenthesis P \rrparenthesis_x]\rfloor_x \equiv \lceil[\langle x \triangleleft \overline{o}\rangle \parr \llparenthesis P \rrparenthesis_x]\rfloor_x
\end{array}
$$

where Lemma 5.3 implies the existence of $\mathscr{D}$. The here above derivation requires q↑ because mixp is derivable in {q↑}.

We simulate s−intro with the following derivation:

$$
\begin{array}{rc}
 & \llparenthesis (M)\,\{x = N\} \rrparenthesis_o \\ \cline{2-2}\cline{2-2}
\mathsf{e}_2 & \lceil \llparenthesis (M)\,\{x = N\} \rrparenthesis_o \rfloor_p \\
 & \mathscr{D}'\Big\Vert \\
 & \lceil[\lceil[\llparenthesis M \rrparenthesis_p \parr \llparenthesis N \rrparenthesis_x]\rfloor_x \parr (p \otimes \overline{o})]\rfloor_p \equiv \lceil[\llparenthesis (M)\,\{x = N\} \rrparenthesis_p \parr (p \otimes \overline{o})]\rfloor_p \\ \hline\hline
\mathsf{e}_1 & \lceil[\lceil[\lceil \llparenthesis M \rrparenthesis_p \rfloor_{p'} \parr \llparenthesis N \rrparenthesis_x]\rfloor_x \parr (p \otimes \overline{o})]\rfloor_p \\
 & \mathscr{D}\Big\Vert \\
 & \lceil[\lceil[\lceil[\llparenthesis M \rrparenthesis_{p'} \parr (p' \otimes \overline{p})]\rfloor_{p'} \parr \llparenthesis N \rrparenthesis_x]\rfloor_x \parr (p \otimes \overline{o})]\rfloor_p \\ \cline{2-2}
\mathsf{r}\downarrow & \lceil[\lceil\lceil[\llparenthesis M \rrparenthesis_{p'} \parr (p' \otimes \overline{p})]\rfloor_{p'}\rfloor_x \parr \lceil \llparenthesis N \rrparenthesis_x \rfloor_x \parr (p \otimes \overline{o})]\rfloor_p \\ \hline\hline
\mathsf{e}_0 & \llparenthesis (\lambda x.M)\, N \rrparenthesis_o \equiv \lceil[\lceil\lceil[\llparenthesis M \rrparenthesis_{p'} \parr (p' \otimes \overline{p})]\rfloor_{p'}\rfloor_x \parr \lceil \llparenthesis N \rrparenthesis_q \rfloor_q \parr (p \otimes \overline{o})]\rfloor_p
\end{array}
$$

where:

- Lemma 5.3 implies the existence of both $\mathscr{D}, \mathscr{D}'$;
- $\lceil \llparenthesis N \rrparenthesis_q \rfloor_q$ in the conclusion of e_0 becomes $\lceil \llparenthesis N \rrparenthesis_q \{^x/_q\}\rfloor_x \approx \lceil \llparenthesis N \rrparenthesis_x \rfloor_x$ in its premise because q only occurs as output channel name in a pair $(p'' \otimes q)$, for some p'', and nowherelse;
- in the conclusion of e_1, the channel p' has disappeared from $\llparenthesis M \rrparenthesis_p$;
- in the conclusion of e_2, the channel p has disappeared from $\llparenthesis (M)\,\{x = N\} \rrparenthesis_o$.

We simulate s−λ with the following derivation:

$$
\begin{array}{rc}
 & \lceil\lceil[\lceil[\llparenthesis M \rrparenthesis_p \parr \llparenthesis P \rrparenthesis_x]\rfloor_x \parr (p \otimes \overline{o})]\rfloor_p\rfloor_y \equiv \lceil\lceil[\llparenthesis (M)\,\{x = P\} \rrparenthesis_p \parr (p \otimes \overline{o})]\rfloor_p\rfloor_y \equiv \llparenthesis \lambda y.(M)\,\{x = P\} \rrparenthesis_o \\ \hline\hline
\mathsf{e}_0 & \lceil\lceil\lceil[\llparenthesis M \rrparenthesis_p \parr (p \otimes \overline{o}) \parr \llparenthesis P \rrparenthesis_x]\rfloor_p\rfloor_y\rfloor_x \\ \cline{2-2}
\mathsf{r}\downarrow,\mathsf{r}\downarrow & \lceil[\lceil\lceil[\llparenthesis M \rrparenthesis_p \parr (p \otimes \overline{o})]\rfloor_p\rfloor_y \parr \lceil\lceil \llparenthesis P \rrparenthesis_x \rfloor_p\rfloor_y]\rfloor_x \\ \hline\hline
 & \llparenthesis (\lambda y.M)\,\{x = P\} \rrparenthesis_o \equiv \lceil[\lceil\lceil[\llparenthesis M \rrparenthesis_p \parr (p \otimes \overline{o})]\rfloor_p\rfloor_y \parr \llparenthesis P \rrparenthesis_x]\rfloor_x
\end{array}
$$

where e_0 applies three of the axioms in Figure 3.

We simulate s − @l with the following derivation:

$$
\begin{array}{rl}
 & \llparenthesis ((M)\,\{x = P\})\, N \rrparenthesis_o \\
 & \equiv \lceil[\llparenthesis (M)\,\{x = P\} \rrparenthesis_p \parr \lceil \llparenthesis N \rrparenthesis_q \rfloor_q \parr (p \otimes \overline{o})]\rfloor_p \\
 & \equiv \lceil[\lceil[\llparenthesis M \rrparenthesis_p \parr \llparenthesis P \rrparenthesis_x]\rfloor_x \parr \lceil \llparenthesis N \rrparenthesis_q \rfloor_q \parr (p \otimes \overline{o})]\rfloor_p \\ \hline\hline
 & \lceil\lceil[\llparenthesis M \rrparenthesis_p \parr \lceil \llparenthesis N \rrparenthesis_q \rfloor_q \parr (p \otimes \overline{o}) \parr \llparenthesis P \rrparenthesis_x]\rfloor_x\rfloor_p \\ \hline\hline
 & \lceil\lceil[\llparenthesis M \rrparenthesis_p \parr \lceil \llparenthesis N \rrparenthesis_q \rfloor_q \parr (p \otimes \overline{o}) \parr \llparenthesis P \rrparenthesis_x]\rfloor_p\rfloor_x \\ \hline
\mathsf{r}\downarrow & \lceil[\lceil[\llparenthesis M \rrparenthesis_p \parr \lceil \llparenthesis N \rrparenthesis_q \rfloor_q \parr (p \otimes \overline{o})]\rfloor_p \parr \lceil \llparenthesis P \rrparenthesis_x \rfloor_p]\rfloor_x \\ \hline\hline
 & \llparenthesis ((M)\, N)\,\{x = P\} \rrparenthesis_o \equiv \lceil[\llparenthesis (M)\, N \rrparenthesis_o \parr \llparenthesis P \rrparenthesis_x]\rfloor_x \equiv \lceil[\lceil[\llparenthesis M \rrparenthesis_p \parr \lceil \llparenthesis N \rrparenthesis_q \rfloor_q \parr (p \otimes \overline{o})]\rfloor_p \parr \llparenthesis P \rrparenthesis_x]\rfloor_x
\end{array}
$$

The simulation of s − @r works like the one of s − @l.