A deep inference system with a self-dual binder which is complete for linear lambda calculus*

LUCA ROVERSI, *Università di Torino — Dipartimento di Informatica*
E-mail: luca.roversi@unito.it

Abstract

We recall that SBV, a proof system developed under the methodology of deep inference, extends multiplicative linear logic with the self-dual non-commutative logical operator Seq. We introduce SBVB that extends SBV by adding the self-dual binder Sdb on names. The system SBVB is consistent because we prove that (the analogous of) cut-elimination holds for it. Moreover, the resulting interplay between Seq and Sdb can model β-reduction of linear λ-calculus inside the cut-free subsystem BVB of SBVB. The long-term aim is to keep developing a programme whose goal is to give pure logical accounts of computational primitives under the proof-search-as-computation analogy, by means of minimal and incremental extensions of SBV.

Keywords: Structural proof-theory, deep inference, linear lambda calculus, calculus of communicating systems.

1 Introduction

This is a work in structural proof-theory. We extend the paradigmatic system of the deep inference methodology, which is SBV [5] and which is conjectured to be the same as pomset logic in [14].

Deep inference: One of the main aspects of deep inference is that logical systems can be designed as they were rewriting systems, i.e. systems with rules that apply *deeply* inside terms or, equivalently, in any context. We must read 'deep' as opposed to 'shallow'. Rules of sequent and natural deduction systems are shallow because they build proofs whose form mimics the one of formulas. The system SBV substantially extends multiplicative linear logic [3] with the non-commutative binary structure Seq. The logical properties of Seq are strictly connected to the expressiveness of SBV. Any limits we might put on the application depth of SBV rules would yield a strictly less expressive system [21]. An extension of SBV by means of linear logic exponentials is SNEL [6–8, 20] whose provability is undecidable [18].

Contributions and motivations: We introduce SBVB. It is SBV plus a *binder* that we identify as Sdb which abbreviates 'Self-dual binder'. Sdb binds variable names of SBVB and can be viewed as a restricted form of a quantifier that we do not need to classify as either existential or universal. This is why Sdb naturally becomes self-dual. So, SBVB can be viewed as a minimal extension of SBV by means of formulas, also called structures, whose instances identify regions where specific variable names can change essentially freely.

Two parts compose this work.

The first is about proving that SBVB is consistent i.e. that it enjoys Splitting (Section 3) which says that the subset BVB of SBVB plays the role of cut-free fragment.

*To Rita Mannarino 1938–2011, and Umberto Spina 1924–2000.

Vol. 26 No. 2, © The Author, 2014. Published by Oxford University Press. All rights reserved.
For Permissions, please email: journals.permissions@oup.com
Published online June 5, 2014 doi:10.1093/logcom/exu033

The second part is about giving Sdb an operational semantics. We show that the interplay between Sdb and Seq makes proof-search inside BVB complete w.r.t. the basic functional computation expressed by linear λ-calculus. We recall that functions linear λ-calculus represents use their arguments exactly once in the course of the evaluation. So, the set of functions it can express is quite limited, yet large enough to let the decision about which is the normal form of two linear λ-terms a *polynomial time complete* problem [11]. Proving the completeness of BVB w.r.t. linear λ-calculus amounts to first defining an embedding $(\!|\_|\!)\_$ from linear λ-terms to formulas of BVB (Section 5.) Then, completeness states that, for every linear λ-term M and every atom o, which plays the role of an output-channel, if M reduces to N then there is a derivation Γ of BVB that derives the formula $(\!|M|\!)_o$ from the formula $(\!|N|\!)_o$ (Theorem 5.5.) For example, let us recall a possible encoding of Boolean negation and of Boolean values as λ-terms:

$$\texttt{Not} \equiv \lambda z.\lambda x.\lambda y.((z)y)x \ , \qquad \texttt{True} \equiv \lambda w.\lambda z.(w)z \ , \qquad \texttt{False} \equiv \lambda w.\lambda z.(z)w \ .$$

Figure 1 shows (part of) a non-trivial example of completeness. The derivation of BVB concludes with the encoding of (Not)True. The premise encodes the corresponding β-reduct $\lambda x.\lambda y.((\texttt{True})y)x$.

Related works: This work improves [16, 17]. It operates simplifications that concern some basic definitions, some terminology and which allow to relate SBVB directly to linear λ-calculus without explicit substitutions, unlike in [16, 17].

With [1, 4, 9] we share the aim of giving a computational interpretation to a deep inference system.

The former restates natural deduction of the negative fragment of intuitionistic logic into a deep inference system from which extracting an algebra of combinators where interpreting λ-terms.

Atomic lambda-calculus, based on a Curry–Howard interpretation of a deep inference proof system for intuitionistic logic, is the subject of [9]. The computational interpretation of the medial-rule of deep inference allows reduction steps to be atomic and to achieve fully lazy sharing.

The latter investigates relations among lambda calculi with explicit substitutions and intuitionistic systems, redefined in accordance with the deep inference approach to proof theory. Specifically, [4] shows the impact of intuitionistic logic, reworked in terms of nested sequents or of calculus of structures, on the design of λ-calculi with explicit substitutions. The investigation proceeds under both the proofs-as-programs and the formulas-as-programs paradigms.

The binder Sdb we introduce to extend SBV shares with the ∇-quantifier of [12] the property of being self-dual. A connection between Sdb and the ∇-quantifier would require to relate the sequent calculus system for generic judgments in [12] with SBVB in the style of [4].

Finally, [22] inspired the two-arguments map $(\!|\_|\!)\_$ from linear λ-terms to formulas of BVB. Anticipating a bit the content of Section 4, the definition of the basic clause of $(\!|\_|\!)\_$ is $(\!|x|\!)_o = \langle x \lhd \overline{o} \rangle$. Intuitively, the linear λ-calculus variable x in $(\!|\_|\!)\_$ becomes the name of an input channel to the left of the occurrence $\lhd$ of Seq. The input channel is forwarded to the output channel o in analogy with the *forwarder* $[x]_o = x(\circ).\overline{o}\langle\circ\rangle$ which comes from [10] and which is one of the defining clauses of the *output-based embedding* of *standard* λ-calculus *with explicit substitutions* into π-calculus [22]. So, SBV can model a forwarder, the basic input/output communication flow that λ-variables realize. More specifically, Sdb allows to model any *on-the-fly renaming* of channels that serves to model the substitution of a term for a bound variable, i.e. the linear β-reduction process of linear λ-calculus.

Organization of the work: Section 2 introduces the extension SBVB (BVB) of SBV (BV). Section 3 proves that SBVB is consistent by extending the proof of (the analogous of) cut-elimination for SBV to SBVB. Section 4 recalls linear λ-calculus and defines the embedding of its terms to formulas of

FIGURE 1. Computing λ-term (`Not`)`True` in BVB.

BVB. Section 5 shows the completeness of BVB w.r.t. linear λ-calculus. Section 6 comments about the lack of a reasonable soundness of BVB w.r.t. λ-calculus and points to future work.

2 Systems SBVB and BVB

We rework the contents of [16, 17].

Structures: Let $a, b, c, \ldots$ denote the elements of a countable set of *positive propositional variables*. Let $\overline{a}, \overline{b}, \overline{c}, \ldots$ denote the elements of a countable set of *negative propositional variables*. The set of *names*, which we range over by $\mathfrak{n}$ and $\mathfrak{m}$, contains positive and negative propositional variables. Let $\circ$ be a constant, different from any name, which we call *unit*. The set of *atoms* contains both names and the unit. The set of *structures* identifies formulas of SBVB. Structures belong to the language of the grammar in (1).

$$R ::= \circ \mid \mathfrak{n} \mid \overline{R} \mid (R \otimes R) \mid \langle R \triangleleft R \rangle \mid [R \parr R] \mid \lceil R \rfloor_a \tag{1}$$

We use P, R, T, U, V to range over structures. The Not structure is $\overline{R}$, the CoPar structure is $(R \otimes T)$, the Seq structure is $\langle R \triangleleft T \rangle$, the Par structure is $[R \parr T]$ and the new *self-dual binder* Sdb is $\lceil R \rfloor_a$. By definition, neither $\lceil R \rfloor_{\overline{a}}$ nor $\lceil R \rfloor_\circ$ belong to the syntax. Sdb induces notions of *free* and *bound occurrences of names*, defined in (2).

$$\begin{array}{ll}
\{a\} = \mathrm{fn}(a) & \emptyset = \mathrm{bn}(a) \\
a \in \mathrm{fn}(\overline{R}) \text{ if } a \in \mathrm{fn}(R) & a \in \mathrm{bn}(\overline{R}) \text{ if } a \in \mathrm{bn}(R) \\
a \in \mathrm{fn}((R \otimes T)) \text{ if } a \in \mathrm{fn}(R) \cup \mathrm{fn}(T) & a \in \mathrm{bn}((R \otimes T)) \text{ if } a \in \mathrm{bn}(R) \cup \mathrm{bn}(T) \\
a \in \mathrm{fn}(\langle R \triangleleft T \rangle) \text{ if } a \in \mathrm{fn}(R) \cup \mathrm{fn}(T) & a \in \mathrm{bn}(\langle R \triangleleft T \rangle) \text{ if } a \in \mathrm{bn}(R) \cup \mathrm{bn}(T) \\
a \in \mathrm{fn}([R \parr T]) \text{ if } a \in \mathrm{fn}(R) \cup \mathrm{fn}(T) & a \in \mathrm{bn}([R \parr T]) \text{ if } a \in \mathrm{bn}(R) \cup \mathrm{bn}(T) \\
a \in \mathrm{fn}(\lceil R \rfloor_b) \text{ if } a \not\equiv b \text{ and } a \in \mathrm{fn}(R) & a \in \mathrm{bn}(\lceil R \rfloor_b) \text{ if } a \equiv b \text{ or } a \in \mathrm{bn}(R)
\end{array} \tag{2}$$

(Structure) Contexts: We denote them by $S\{\ \}$. A context is a structure with a single hole $\{\ \}$ in it which represents a generic but unknown structure. 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.

Congruence $\approx$ among structures: The definition of the congruence requires an operation of *renaming* that applies to structures and names. We introduce renaming in (3).

$$\begin{array}{rcl@{\qquad}rcl}
\circ\{^a/_b\} & \equiv & \circ & \langle R \triangleleft T \rangle\{^a/_b\} & \equiv & \langle R\{^a/_b\} \triangleleft T\{^a/_b\} \rangle \\
b\{^a/_b\} & \equiv & a & [R \parr T]\{^a/_b\} & \equiv & [R\{^a/_b\} \parr T\{^a/_b\}] \\
c\{^a/_b\} & \equiv & c \text{ with } c \not\equiv b & \lceil R \rfloor_b\{^a/_b\} & \equiv & \lceil R \rfloor_b \\
\overline{R}\{^a/_b\} & \equiv & \overline{R\{^a/_b\}} & \lceil R \rfloor_c\{^a/_b\} & \equiv & \lceil R\{^a/_b\} \rfloor_c \text{ with } c \not\equiv b \\
(R \otimes T)\{^a/_b\} & \equiv & (R\{^a/_b\} \otimes T\{^a/_b\}) & & &
\end{array} \tag{3}$$

So, $R\{^a/_b\}$ means that we eventually simultaneously replace the name a for every free occurrence of the name b and the name $\overline{a}$ for every free occurrence of the name $\overline{b}$ inside R.

By definition, structures are partitioned by the smallest congruence $\approx$ we obtain as reflexive, symmetric, transitive and contextual closure of the relation $\sim$ whose defining clauses are (4), through (20).

Negation

$$\overline{\circ} \sim \circ \quad (4)$$
$$\overline{\overline{R}} \sim R \quad (5)$$
$$\overline{[R \parr T]} \sim (\overline{R} \otimes \overline{T}) \quad (6)$$
$$\overline{(R \otimes T)} \sim [\overline{R} \parr \overline{T}] \quad (7)$$
$$\overline{\langle R \triangleleft T \rangle} \sim \langle \overline{R} \triangleleft \overline{T} \rangle \quad (8)$$
$$\overline{\lceil R \rfloor_a} \sim \lceil \overline{R} \rfloor_a \quad (9)$$

Commutativity

$$[R \parr T] \sim [T \parr R] \quad (10)$$
$$(R \otimes T) \sim (T \otimes R) \quad (11)$$

Associativity

$$(R \otimes (T \otimes V)) \sim ((R \otimes T) \otimes V) \quad (12)$$
$$\langle R \triangleleft \langle T \triangleleft V \rangle \rangle \sim \langle \langle R \triangleleft T \rangle \triangleleft V \rangle \quad (13)$$
$$[R \parr [T \parr V]] \sim [[R \parr T] \parr V] \quad (14)$$

Unit

$$(\circ \otimes R) \sim R \quad (15)$$
$$\langle \circ \triangleleft R \rangle \sim \langle R \triangleleft \circ \rangle \sim R \quad (16)$$
$$[\circ \parr R] \sim R \quad (17)$$

α-rule

$$\lceil R \rfloor_a \sim R \quad \text{if } a \notin \mathrm{fn}(R) \quad (18)$$
$$\lceil R\{^a/_b\} \rfloor_a \sim \lceil R \rfloor_b \quad \text{if } a \in \mathrm{fn}(R) \quad (19)$$
$$\lceil \lceil R \rfloor_b \rfloor_a \sim \lceil \lceil R \rfloor_a \rfloor_b \quad (20)$$

Contextual closure means that $S\{R\} \approx S\{T\}$ whenever $R \approx T$. We remark that Sdb is self-dual like Seq is. When introducing the logical rules we shall clarify why. The axiom (19) says that a and $\overline{a}$ rename, i.e. replace, b and $\overline{b}$, respectively, exactly when $b \in \mathrm{fn}(R)$. The axiom (20) allows to abbreviate $\lceil \cdots \lceil R \rfloor_{a_1} \cdots \rfloor_{a_n}$ as $\lceil R \rfloor_{\vec{a}}$ where $\vec{a}$ is one of the permutations of $a_1, \ldots, a_n$. Congruence extends naturally to contexts. For example, $(S\{\ \} \otimes R) \approx (R \otimes S\{\ \})$. Finally, we say that R is a *substructure* of T whenever $S\{R\} \approx T$, for some $S\{\ \}$.

Structures which are Bound-variable equivalent: We say that two structures R and T are bound-variable equivalent whenever there exist two structures U and V such that $R \approx U$ and $V \approx T$ and $\mathrm{bn}(U) = \mathrm{bn}(V)$. For example, $[a \parr \lceil b \rfloor_b]$ and $\langle \lceil \overline{d} \rfloor_d \triangleleft d \triangleleft c \rangle$ are bound-variable equivalent. Instead a and $\lceil \overline{d} \rfloor_d$ are not.

Free and bound names in contexts: Sdb induces free and bound names also on contexts as given in (21) and (22), respectively.

$$
\begin{array}{l}
\emptyset = \mathrm{fn}(\{\ \}) \\
a \in \mathrm{fn}(\overline{S\{\ \}}) \text{ if } a \in \mathrm{fn}(S\{\ \}) \\
a \in \mathrm{fn}((S\{\ \} \otimes R)) \text{ if } a \in \mathrm{fn}(S\{\ \}) \cup \mathrm{fn}(R) \\
a \in \mathrm{fn}(\langle S\{\ \} \triangleleft R\rangle) \text{ if } a \in \mathrm{fn}(S\{\ \}) \cup \mathrm{fn}(R) \\
a \in \mathrm{fn}(\langle R \triangleleft S\{\ \}\rangle) \text{ if } a \in \mathrm{fn}(S\{\ \}) \cup \mathrm{fn}(R) \\
a \in \mathrm{fn}([S\{\ \} \parr R]) \text{ if } a \in \mathrm{fn}(S\{\ \}) \cup \mathrm{fn}(R) \\
a \in \mathrm{fn}(\lceil S\{\ \} \rfloor_b) \text{ if } a \not\equiv b \text{ and } a \in \mathrm{fn}(S\{\ \})
\end{array}
\tag{21}
$$

$$
\begin{array}{l}
\emptyset = \mathrm{bn}(\{\ \}) \\
a \in \mathrm{bn}(\overline{S\{\ \}}) \text{ if } a \in \mathrm{bn}(S\{\ \}) \\
a \in \mathrm{bn}((S\{\ \} \otimes R)) \text{ if } a \in \mathrm{bn}(S\{\ \}) \cup \mathrm{bn}(T) \\
a \in \mathrm{bn}(\langle S\{\ \} \triangleleft R\rangle) \text{ if } a \in \mathrm{bn}(S\{\ \}) \cup \mathrm{bn}(T) \\
a \in \mathrm{bn}(\langle R \otimes S\{\ \}\rangle) \text{ if } a \in \mathrm{bn}(S\{\ \}) \cup \mathrm{bn}(T) \\
a \in \mathrm{bn}([S\{\ \} \parr R]) \text{ if } a \in \mathrm{bn}(S\{\ \}) \cup \mathrm{bn}(T) \\
a \in \mathrm{bn}(\lceil S\{\ \} \rfloor_b) \text{ if } a \equiv b \text{ or } a \in \mathrm{bn}(S\{\ \})
\end{array}
\tag{22}
$$

Size of the structures and of the contexts: The *size* $|R|$ of a structure R is in (23). It counts the number of occurrences of names in R plus the number of occurrences of Sdb that effectively bind a name.

$$
\begin{array}{ll}
|\mathfrak{n}| = 1 & |\circ| = 0 \\
|[R \parr T]| = |R| + |T| & |\overline{R}| = |R| \\
|\langle R \triangleleft T\rangle| = |R| + |T| & |\lceil R \rfloor_a| = |R| \text{ if } a \notin \mathrm{fn}(R) \\
|(R \otimes T)| = |R| + |T| & |\lceil R \rfloor_a| = |R| + 1 \text{ if } a \in \mathrm{fn}(R)
\end{array}
\tag{23}
$$

EXAMPLE 2.1 (Size of the structures)
We have $|[a \parr \overline{a}]| = |\lceil [a \parr \overline{a}] \rfloor_b| = 2$ for we do not count the occurrence of $\lceil \_ \rfloor\_$. Instead, we count it in $\lceil [a \parr \overline{a}] \rfloor_a$, getting $|\lceil [a \parr \overline{a}] \rfloor_a| = 3$. ■

The size $|S\{\ \}|$ of the context $S\{\ \}$ is in (24). It looks at $\{\ \}$ as it was $\circ$.

$$
\begin{array}{ll}
|\{\ \}| = 0 & |\langle R \triangleleft S\{\ \}\rangle| = |S\{\ \}| + |R| \\
|\overline{S\{\ \}}| = |S\{\ \}| & |[S\{\ \} \parr R]| = |S\{\ \}| + |R| \\
|(S\{\ \} \otimes R)| = |S\{\ \}| + |R| & |\lceil S\{\ \} \rfloor_a| = |S\{\ \}| \text{ if } a \notin \mathrm{fn}(S\{\ \}) \\
|\langle S\{\ \} \triangleleft R\rangle| = |S\{\ \}| + |R| & |\lceil S\{\ \} \rfloor_a| = |S\{\ \}| + 1 \text{ if } a \in \mathrm{fn}(S\{\ \})
\end{array}
\tag{24}
$$

REMARK 2.2 (The size may change if names get bound)
In general, $|S\{\ \}|+|R| \neq |R|$. For example, $|\lceil\{\ \}\rfloor_a|+|a| = |\{\ \}|+|a| = 0+1 = 1$ differs from $|\lceil a\rfloor_a|+|a| = 1+1 = 2$. ■

The system SBVB: It contains the set of inference rules defined in (25). Every rule has form $\rho\dfrac{T}{R}$ where ρ is the *name of the rule*, T is its *premise* and R is its *conclusion*.

$$\begin{array}{ccccc}
 & \mathsf{ai}{\downarrow}\dfrac{\circ}{[a \parr \overline{a}]} & & & \mathsf{ai}{\uparrow}\dfrac{(a \otimes \overline{a})}{\circ} \\
\mathsf{q}{\downarrow}\dfrac{\langle [R \parr U] \lhd [T \parr V]\rangle}{[\langle R \lhd T\rangle \parr \langle U \lhd V\rangle]} & & \mathsf{s}\dfrac{([R \otimes T] \parr U)}{[(R \parr U) \otimes T]} & & \mathsf{q}{\uparrow}\dfrac{(\langle R \lhd T\rangle \otimes \langle U \lhd V\rangle)}{\langle (R \otimes U) \lhd (T \otimes V)\rangle} \\
 & \mathsf{u}{\downarrow}\dfrac{\lceil [R \parr U]\rfloor_a}{[\lceil R\rfloor_a \parr \lceil U\rfloor_a]} & & & \mathsf{u}{\uparrow}\dfrac{(\lceil R\rfloor_a \otimes \lceil U\rfloor_a)}{\lceil (R \otimes U)\rfloor_a}
\end{array} \tag{25}$$

Every instance of an inference rule $\rho\dfrac{T}{R}$ can be used in any context $S\{\ \}$, i.e. as $\rho\dfrac{S\{T\}}{S\{R\}}$. This means that if a structure U matches $S\{R\}$ then $S\{R\}$ can be rewritten to $S\{T\}$. This justifies calling R the *redex* of ρ and T its *reduct*.

***Down* and *up* fragments of SBVB:** The set $\{\mathsf{ai}{\downarrow}, \mathsf{s}, \mathsf{q}{\downarrow}, \mathsf{u}{\downarrow}\}$ is the *down fragment* BVB of SBVB. The *up fragment* is $\{\mathsf{ai}{\uparrow}, \mathsf{s}, \mathsf{q}{\uparrow}, \mathsf{u}{\uparrow}\}$. So s belongs to both. The rule $\mathsf{ai}{\uparrow}$ *plays the role of the cut rule of sequent calculus*. The down rule for Sdb restricts the following one:

$$\mathsf{u}{\downarrow}\frac{\forall a.[R \parr U]}{[\forall a.R \parr \exists a.U]}\ ,$$

given in [19], to bind variable names only. Limiting Sdb to bind variables implies that the difference between existentially and universally quantified names disappears. The reason is that the cut-elimination will have no need to distinguish between the substitution of an existentially quantified variable for a universally quantified one, or vice versa. So, Sdb becomes self-dual.

Derivations versus proofs: A *derivation* in SBVB is either a structure, considered up to the equivalence relation $\approx$, or a chain of consecutive instances of the rules in (25). Both Γ and Δ will range over derivations. The topmost structure in a derivation is its *premise*. The *conclusion* is its bottommost structure. A derivation Γ of a structure R in SBVB from a structure T in SBVB which only uses a subset B of rules in SBVB is denoted by $\Gamma \dfrac{T}{R}\!\!\parallel_{\mathsf{B}}$ or, equivalently, by $\Gamma : T \vdash_{\mathsf{B}} R$.

The derivation $\Gamma : T \vdash_{\mathsf{B}} R$ is a *proof* whenever $T \approx \circ$. We denote proofs by $\Phi \dfrac{\circ}{R}\!\!\parallel_{\mathsf{B}}$ or $\dfrac{\Phi}{R}\!\!\top_{\mathsf{B}}$ or $\Phi : \vdash_{\mathsf{B}} R$.

Both Φ, and Ψ will range over proofs. We will drop B when clear from the context.

We will contract parts of derivations by means of *macro rules*. A macro rule takes the form $\{\rho_1,\ldots,\rho_m,n_1,\ldots,n_p\}\dfrac{T}{\!\!=\!\!R}$. It highlights that R derives from T by using instances of the rules $\rho_1,\ldots,\rho_m$ not

necessarily in the given order, arbitrarily interspersed with p instances $n_1, \ldots, n_p$ of the equivalences (4), …, (20). In a macro rule, if $q > 1$ instances of some axiom (n) of (4), …, (20), occurs among $n_1, \ldots, n_p$, then we write $(n)^q$.

EXAMPLE 2.3
Two equivalent ways to contract a given derivation into a macro rule are:

$$\{\mathsf{s},(11),\mathsf{s},(11)\}\frac{\lceil [R \parr U_1] \otimes [T \parr U_2] \rfloor_a}{\lceil [(R \otimes T) \parr U_1 \parr U_2] \rfloor_a} \quad , \quad \{\mathsf{s}^2,(11)^2\}\frac{\lceil ([R \parr U_1] \otimes [T \parr U_2]) \rfloor_a}{\lceil [(R \otimes T) \parr U_1 \parr U_2] \rfloor_a} \quad .$$

The leftmost one highlights the order of application of rules and equivalences. ■

REMARK 2.4
When writing structures we tend to omit the use of parenthesis, always pushing negation towards atoms. For example, we prefer $[[a \parr b] \parr \overline{[\circ \parr c]}]$ to $[a \parr b \parr (\circ \otimes \overline{c})]$. ■

Length of a derivation: Let B any subset of SBVB. The *length* $\|\Gamma\|$ of a derivation $\Gamma : T \vdash_{\mathsf{B}} R$ is the number of rule instances in Γ. So, $\|\Gamma\|$ does not count the application of any instance of (4), …, (20).

Admissible and derivable rules: Let B be a subset of rules in SBVB. A rule ρ is *admissible* for B if $\rho \notin \mathsf{B}$ and, for every derivation Γ such that $\Gamma : T \vdash_{\{\rho\} \cup \mathsf{B}} R$, there is $\Gamma' : T \vdash_{\mathsf{B}} R$. A rule ρ is *derivable* in B if $\rho \notin \mathsf{B}$ and, for every instance $\rho\dfrac{T}{R}$, there exists a derivation Γ in B such that $\Gamma : T \vdash_{\mathsf{B}} R$.

Basic properties of BVB: In (26) we recall relevant rules derivable in SBV, hence in SBVB.

$$\mathsf{i}{\downarrow}\frac{\circ}{[R \parr \overline{R}]} \qquad \mathsf{t}{\downarrow}\frac{S\langle R \lhd T\rangle}{[S\langle R \lhd \overline{a}\rangle \parr \langle a \lhd T\rangle]}(*)$$

$$\mathsf{i}{\uparrow}\frac{(R \otimes \overline{R})}{\circ} \qquad \mathsf{t}{\uparrow}\frac{(S\langle R \lhd \overline{a}\rangle \otimes \langle a \lhd T\rangle)}{S\langle R \lhd T\rangle}(*) \tag{26}$$

$$(*) \text{ requires } (\{a\} \cup \mathrm{fn}(T)) \cap \mathrm{bn}(S\{\ \}) = \emptyset.$$

General interaction up is $\mathsf{i}{\uparrow}$ in (26). It is derivable in $\{\mathsf{ai}{\uparrow}, \mathsf{s}, \mathsf{q}{\uparrow}, \mathsf{u}{\uparrow}\}$, reasoning by induction on $|R|$ and proceeding by case analysis on the form of R. Few steps of the proof relative to the case with $R \approx \lceil T \rfloor_a$ follow:

$$\cfrac{\cfrac{\cfrac{\cfrac{(\lceil T \rfloor_a \otimes \overline{\lceil T \rfloor_a})}{\{(9)\}=\!=}}{\mathsf{u}{\uparrow}}}{\mathsf{i}{\uparrow}}}{\{(18)\}=\!=} \quad \begin{array}{l} (\lceil T \rfloor_a \otimes \lceil \overline{T} \rfloor_a) \\ \lceil (T \otimes \overline{T}) \rfloor_a \\ \text{ind. hypothesis} \\ \circ \end{array} .$$

Similar arguments which we know from SBV apply to Not, CoPar, Seq and Par. Symmetrically, *general interaction down* $\mathsf{i}{\downarrow}$ is derivable in $\{\mathsf{ai}{\downarrow}, \mathsf{s}, \mathsf{q}{\downarrow}, \mathsf{u}{\downarrow}\}$.

The next proposition shows that two structures R and T of BVB can be moved one aside the other no matter what the context is. It is a standard but basic result in the calculus of structures which smoothly extends to the binder Sdb of BVB.

PROPOSITION 2.5 (Context permeability)
Let $S\{\ \}$ be any context and R, T be any structures. Then $S[R \parr T] \vdash_{\{\mathsf{q}\downarrow,\mathsf{u}\downarrow,\mathsf{s}\}} [S\{R\} \parr T]$.

PROOF. By induction on $|S\{\ \}|$, proceeding by case analysis on the form of $S\{\ \}$.

1. Let $S\{\ \} \approx \{\ \}$. The statement holds because $S[R \parr T] \approx [S\{R\} \parr T] \approx [R \parr T]$ and $[R \parr T]$ is a structure, hence, by definition, a derivation.
2. Let $S\{\ \} \approx \langle S'\{\ \} \lhd U\rangle$. Then:

$$\begin{array}{c} \langle S'[R \parr T] \lhd U\rangle \\ \Gamma \Big\| \\ \{\mathsf{q}\downarrow,(16)^2\}\dfrac{\langle [S'\{R\} \parr T] \lhd U\rangle}{[\langle S'\{R\} \lhd U\rangle \parr T]} \end{array},$$

   where Γ exists by inductive hypothesis, which holds because $|S'\{\ \}| < |S\{\ \}|$.
   The case where $S\{\ \} \approx (S'\{\ \} \otimes U)$ is analogous using s in place of $\mathsf{q}\downarrow$ and (15) in place of (16).
3. Let $S\{\ \} \approx \lceil S'\{\ \} \rfloor_a$. Without loss of generality, by (19), we can assume $a \notin \mathrm{fn}(T)$. Then:

$$\begin{array}{c} \lceil S'[R \parr T] \rfloor_a \\ \Gamma \Big\| \\ \mathsf{u}\downarrow\dfrac{\lceil [S'\{R\} \parr T] \rfloor_a}{\{(18)\}\dfrac{[\lceil S'\{R\} \rfloor_a \parr \lceil T \rfloor_a]}{[\lceil S'\{R\} \rfloor_a \parr T]}} \end{array},$$

   where Γ exists by inductive hypothesis, which holds because $|S'\{\ \}| < |S\{\ \}|$. ■

FACT 2.6
The rule Seq-*transitive down* $\mathsf{t}\downarrow$ in (26) is derivable in BVB.

PROOF. The following derivation:

$$\begin{array}{c} S\langle R \lhd T\rangle \\ \{\mathsf{ai}\downarrow,(17),(16)\}\dfrac{}{S\langle R \lhd \langle [\overline{a} \parr a] \lhd [\circ \parr T]\rangle\rangle} \\ \mathsf{q}\downarrow\dfrac{}{S\langle R \lhd [\langle \overline{a} \lhd \circ\rangle \parr \langle a \lhd T\rangle]\rangle} \\ \{(17),(16)\}\dfrac{}{S\langle [R \parr \circ] \lhd [\overline{a} \parr \langle a \lhd T\rangle]\rangle} \\ \mathsf{q}\downarrow\dfrac{}{S[\langle R \lhd \overline{a}\rangle \parr \langle \circ \lhd \langle a \lhd T\rangle\rangle]} \\ \{(16)\}\dfrac{}{S[\langle R \lhd \overline{a}\rangle \parr \langle a \lhd T\rangle]} \\ \Gamma \Big\| \\ [S\langle R \lhd \overline{a}\rangle \parr \langle a \lhd T\rangle] \end{array},$$

derives $\mathsf{t}\downarrow$, where Γ exists by applying Proposition 2.5. ■

The following fact, which we can prove by inspecting the behaviour of the rules in BVB and the definition of $\approx$, highlights that Sdb is a binder.

FACT 2.7
Let a be a name, and let U, V be structures.

1. If $\Gamma : V \vdash_{\mathsf{BVB}} \lceil U \rfloor_a$ then there exist R and Γ' such that $\Gamma' : \lceil R \rfloor_a \vdash_{\mathsf{BVB}} \lceil U \rfloor_a$ and $\lceil R \rfloor_a \approx V$.
2. Let R be any structure. Then $\Gamma : \lceil R \rfloor_a \vdash_{\mathsf{BVB}} \lceil U \rfloor_a$ if, and only if, $\Gamma' : R \vdash_{\mathsf{BVB}} U$, for some Γ and Γ'.

Finally, no new variable can be introduced in the course of a derivation.

PROPOSITION 2.8 (BVB is affine)
In every $\Gamma : T \vdash_{\mathsf{BVB}} R$, we have $|R| \geq |T|$.

PROOF. By induction on $\|\Gamma\|$, proceeding by case analysis on the lowermost occurrence of inference rule in Γ. ■

3 The systems SBVB and BVB are equivalent

We show that every up rule of SBVB is admissible for BVB which means that SBVB and BVB are equivalent. Since the atomic interaction up rule ai↑ belongs to SBVB and ai↑ plays the role of a cut rule, proving that SBVB is equivalent to BVB amounts to showing that a strong version of cut-elimination holds.

Our proof of equivalence follows the presentation in [8], where the proof relies on the two separate phases *splitting* and *context reduction*. The second phase depends on the first one.

3.1 Splitting

We refer the reader to [5] for an intuitive explanation of splitting.

LEMMA 3.1 (Splitting for Seq and CoPar)
Let R, T, and P be structures. Let a be a name.

1. If $\Phi : \vdash_{\mathsf{BVB}} [\langle R \triangleleft T \rangle \parr P]$ then, for some structures P_1 and P_2, there exist the derivation $\Gamma : \langle P_1 \triangleleft P_2 \rangle \vdash_{\mathsf{BVB}} P$ and the proofs $\Phi_1 : \vdash_{\mathsf{BVB}} [R \parr P_1]$ and $\Phi_2 : \vdash_{\mathsf{BVB}} [T \parr P_2]$.
2. If $\Phi : \vdash_{\mathsf{BVB}} [(R \otimes T) \parr P]$ then, for some structures P_1 and P_2, there exist the derivation $\Gamma : [P_1 \parr P_2] \vdash_{\mathsf{BVB}} P$ and the proofs $\Phi_1 : \vdash_{\mathsf{BVB}} [R \parr P_1]$ and $\Phi_2 : \vdash_{\mathsf{BVB}} [T \parr P_2]$.

PROOF. The proof of the two statements is a simultaneous induction on the lexicographic order of the pair $(|V|, \|\Phi\|)$, where V is one between $[\langle R \triangleleft T \rangle \parr P]$ or $[(R \otimes T) \parr P]$. We can proceed by case analysis on how the lowermost rule ρ of Φ operates on $\langle R \triangleleft T \rangle$ or $(R \otimes T)$. We refer to [8] for the details relative to the cases where $\rho \in \{\mathsf{ai}{\downarrow}, \mathsf{q}{\downarrow}, \mathsf{s}\}$.

Point 1 Let ρ be $\mathsf{u}{\downarrow}$. The most interesting case presents $P \approx \lceil U \rfloor_a$, for some U and a, such that $a \in \mathrm{fn}(U)$ and $a \notin \mathrm{fn}(\langle R \triangleleft T \rangle)$. By (18), $\langle R \triangleleft T \rangle \approx \lceil \langle R \triangleleft T \rangle \rfloor_a$ and Φ is:

$$\mathsf{u}{\downarrow}\frac{\Phi' \Big\Vert \mathsf{BVB}}{\dfrac{\lceil [\langle R \triangleleft T \rangle \parr U] \rfloor_a}{[\lceil \langle R \triangleleft T \rangle \rfloor_a \parr \lceil U \rfloor_a]}} .$$

Point 2 of Fact 2.7 implies that $\Phi'' : \vdash_{\mathsf{BVB}} [\langle R \triangleleft T \rangle \parr U]$ exists. The inductive hypothesis holds on Φ'' because $|[\langle R \triangleleft T \rangle \parr U]| < |[\lceil \langle R \triangleleft T \rangle \rfloor_a \parr \lceil U \rfloor_a]|$. It implies that $\Delta : \langle P_1 \triangleleft P_2 \rangle \vdash_{\mathsf{BVB}} U$

and $\Psi_1 : \vdash_{\mathsf{BVB}} [R \parr P_1]$ and $\Psi_2 : \vdash_{\mathsf{BVB}} [T \parr P_2]$ exist. From Δ we can build $\lceil \langle P_1 \lhd P_2 \rangle \rfloor_a \vdash_{\mathsf{BVB}} \lceil U \rfloor_a$ by applying Point 2.7 of Fact 2.7.

Point 2 The proof of the second statement proceeds analogously, by replacing $(R \otimes T)$ for $\langle R \lhd T \rangle$. ■

LEMMA 3.2 (Splitting for Sdb)
If $\Phi : \vdash [\lceil R \rfloor_a \parr P]$ then, for some structure T, there exist the derivation $\Gamma : \lceil T \rfloor_a \vdash_{\mathsf{BVB}} P$ and the proof $\Psi : \vdash_{\mathsf{BVB}} [R \parr T]$.

PROOF. The proof is by induction on the lexicographic order of the pair $(|[\lceil R \rfloor_a \parr P]|, \|\Phi\|)$. We proceed by case analysis on how the lowermost rule ρ of Φ operates on $\lceil R \rfloor_a$. Since $\mathsf{u}{\downarrow}$ essentially behaves like the modal rule $\mathsf{p}{\downarrow}$ of system NEL in [8], we follow the same proof pattern as given in [8].

1. Let ρ be such that its reduct is completely internal to R or such that it does not alter $\lceil R \rfloor_a$ from its premise to its conclusion. The inductive argument applies obviously.
2. Let ρ be $\mathsf{u}{\downarrow}$ with $[\lceil R \rfloor_a \parr P]$ as redex, where $a \in \mathrm{fn}(R)$ and $P \approx \lceil T \rfloor_a$ such that $a \in \mathrm{fn}(T)$. Then $\Gamma : \lceil T \rfloor_a \vdash_{\mathsf{BVB}} \lceil T \rfloor_a$. Since Φ is:
$$\mathsf{u}{\downarrow} \frac{\begin{array}{c}\Phi' \\ \lceil [R \parr T] \rfloor_a \end{array} \mathsf{BVB}}{[\lceil R \rfloor_a \parr \lceil T \rfloor_a]} .$$
we can conclude by applying Point 2.7 of Fact 2.7 to Φ'.
3. Let ρ be $\mathsf{u}{\downarrow}$. Two remaining cases exist. The first case has $[\lceil R \rfloor_a \parr P]$ as redex with $a \notin \mathrm{fn}(R)$. So, it must be $P \approx \lceil T \rfloor_a$ with $a \in \mathrm{fn}(T)$ otherwise there would no argument to apply $\mathsf{u}{\downarrow}$. Symmetrically, the second case has $[\lceil R \rfloor_a \parr P]$ as redex where $a \in \mathrm{fn}(R)$ and $P \approx \lceil T \rfloor_a$ such that $a \notin \mathrm{fn}(T)$. In both cases we can proceed as in step 3.1 here above. ■

LEMMA 3.3 (Splitting for names)
Let $\Phi : \vdash_{\mathsf{BVB}} [\mathfrak{n} \parr P]$. Then there exists $\Gamma : \overline{\mathfrak{n}} \vdash_{\mathsf{BVB}} P$.

PROOF. We recall the proof from [8] which is by induction on $\|\Phi\|$. We proceed by case analysis on how the lowermost rule ρ of Φ operates on its conclusion.

1. Let ρ be such that its reduct is completely internal to P. The inductive argument applies obviously.
2. Let ρ be $\mathsf{ai}{\downarrow}$. So, it must be $P \approx [\overline{\mathfrak{n}} \parr U]$, for some U. Then Φ is:
$$\{\mathsf{ai}{\downarrow}, (17)\} \frac{\begin{array}{c}\Phi' \\ U \end{array} \mathsf{BVB}}{[\mathfrak{n} \parr \overline{\mathfrak{n}} \parr U]} .$$
Since the proof $\Phi' : \vdash_{\mathsf{BVB}} U$ exists, it exists $\Gamma : \overline{\mathfrak{n}} \vdash_{\mathsf{BVB}} [\overline{\mathfrak{n}} \parr U]$ as well. ■

3.2 *Context reduction*

The goal is to reduce a problem that involves an arbitrary deep context $S\{\ \}$ to a problem that concerns only a shallow context $[\{\ \} \parr P]$.

PROPOSITION 3.4 (Context Reduction in BVB)
Let R be a structure, and $S\{\ \}$ be a context such that $\Phi\colon \vdash_{\mathsf{BVB}} S\{R\}$. There are a structure U and a, possibly empty, sequence of variables $\vec{b}$ such that, for every V which is bound-variable equivalent with R, we have $\Gamma\colon \lceil [V \parr U] \rfloor_{\vec{b}} \vdash_{\mathsf{BVB}} S\{V\}$ and $\Psi\colon \vdash_{\mathsf{BVB}} [R \parr U]$.

PROOF. The proof is by induction on $|S\{\ \}|$, proceeding by case analysis on the form of $S\{\ \}$. We recall the following steps 1, 2 and 3 from [8]. The fourth one is new.

1. Let $S\{R\}$ be R. We can conclude because $\vec{b}$ is the empty sequence and U is $\circ$.
2. Let $S\{R\} \approx [(S'\{R\} \otimes T) \parr P]$ with $T \not\approx \circ$. The assumption is $\Phi\colon \vdash_{\mathsf{BVB}} [(S'\{R\} \otimes T) \parr P]$. Lemma 3.1 implies that there exist P_1, P_2 such that $\Gamma\colon [P_1 \parr P_2] \vdash_{\mathsf{BVB}} P$ and $\Phi_1\colon \vdash_{\mathsf{BVB}} [S'\{R\} \parr P_1]$ and $\Phi_2\colon \vdash_{\mathsf{BVB}} [T \parr P_2]$. The inductive hypothesis holds on Φ_1 because $|[S'\{R\} \parr P_1]| < |[(S'\{R\} \otimes T) \parr P]|$. There are U and $\vec{b}$ such that, for every V which is bound-variable equivalent to R, we have $\Gamma'\colon \lceil [V \parr U] \rfloor_{\vec{b}} \vdash_{\mathsf{BVB}} [S'\{V\} \parr P_1]$ and $\Phi'''\colon \vdash_{\mathsf{BVB}} [R \parr U]$. The proof we are looking for is Φ'''. To build the derivation we fix V bound-variable equivalent R and we use Γ', Φ_2 and Γ as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\lceil [V \parr U] \rfloor_{\vec{b}}}{\Gamma' \Big\| \mathsf{BVB}}}{\{(15)\}\dfrac{[S'\{V\} \parr P_1]}{(\circ \otimes [S'\{V\} \parr P_1])}}}{\Phi_2 \Big\| \mathsf{BVB}}}{\{(11),\mathsf{s}\}\dfrac{([T \parr P_2] \otimes [S'\{V\} \parr P_1])}{[([S'\{V\} \parr P_1] \otimes T) \parr P_2]}}}{\mathsf{s}\dfrac{}{[[(S'\{V\} \otimes T) \parr P_1] \parr P_2]}}}{\{(14)\}\dfrac{}{[(S'\{V\} \otimes T) \parr [P_1 \parr P_2]]}}}{\Gamma \Big\| \mathsf{BVB}}}{[(S'\{V\} \otimes T) \parr P]}$$

3. Let $S\{R\} \approx [\langle S'\{R\} \triangleleft P' \rangle \parr P]$ or $S\{R\} \approx [\langle P' \triangleleft S'\{R\} \rangle \parr P]$. We can proceed as in the previous case.
4. Let $S\{R\} \approx [\lceil S'\{R\} \rfloor_a \parr P]$ with $a \in \mathrm{fn}(S'\{R\})$. Otherwise, it would be meaningless to assume $S\{R\}$ as such. The assumption is $\Phi\colon \vdash_{\mathsf{BVB}} [\lceil S'\{R\} \rfloor_a \parr P]$. Lemma 3.2 implies that T exists such that $\Gamma\colon \lceil T \rfloor_a \vdash_{\mathsf{BVB}} P$, and $\Phi'\colon \vdash_{\mathsf{BVB}} [S'\{R\} \parr T]$. The inductive hypothesis holds on Φ' because $|[S'\{R\} \parr T]| < |[\lceil S'\{R\} \rfloor_a \parr P]|$. There are U and $\vec{b}$ such that, for every V bound-variable equivalent with R, we have $\Gamma'\colon \lceil [V \parr U] \rfloor_{\vec{b}} \vdash_{\mathsf{BVB}} [S'\{V\} \parr T]$ and $\Phi''\colon \vdash_{\mathsf{BVB}} [R \parr U]$. The proof we are looking for is Φ''. To get the derivation we fix V bound-variable equivalent to R and we use Γ and Γ' as follows:

$$\dfrac{\dfrac{\dfrac{\lceil [V \parr U] \rfloor_{b_1,\ldots,b_n,a}}{\Gamma' \Big\| \mathsf{BVB}}}{\mathsf{u}\downarrow\dfrac{\lceil [S'\{V\} \parr T] \rfloor_a}{[\lceil S'\{V\} \rfloor_a \parr \lceil T \rfloor_a]}}}{\Gamma \Big\| \mathsf{BVB}}}{[\lceil S'\{V\} \rfloor_a \parr P]} \ .$$

■

REMARK 3.5 (Condition in the statement of Context Reduction)
Context reduction requires that V and R are bound-variable equivalent because, otherwise, V and R would interact differently with $S\{\ \}$.

For example, let $R \approx \lceil [a \parr b] \rfloor_b$ and $S\{\ \} \approx [\{\ \} \parr \lceil (\overline{a} \otimes \overline{c}) \rfloor_c]$. Let us assume that $\Phi : \vdash_{\mathsf{BVB}} S\{R\}$. We can build:

$$\begin{array}{c} \lceil [[a \parr b] \parr (\overline{a} \otimes \overline{b})] \rfloor_b \\ \Big\Vert \mathsf{BVB} \\ [R \parr \lceil (\overline{a} \otimes \overline{c}) \rfloor_c] \end{array} \quad \text{and} \quad \begin{array}{c} \Big\Vert \mathsf{BVB} \\ [[a \parr b] \parr (\overline{a} \otimes \overline{b})] \end{array},$$

where $(\overline{a} \otimes \overline{b})$ plays the role of U.

Let us replace R by $V \approx \lceil [a \parr c] \rfloor_b$ in Γ which is not bound-variable equivalent with R. The derivation:

$$\begin{array}{c} \lceil [[a \parr c] \parr (\overline{a} \otimes \overline{b})] \rfloor_b \\ \Big\Vert \mathsf{BVB} \\ [V \parr \lceil (\overline{a} \otimes \overline{c}) \rfloor_c] \end{array},$$

would exists, but not the proof $\begin{array}{c} \Big\Vert \mathsf{BVB} \\ [[a \parr c] \parr (\overline{a} \otimes \overline{b})] \end{array}$. ■

3.3 *The Up fragment is admissible for* BVB

We are going to show a modular cut-elimination for SBVB which means we can prove that every rule $\{\mathsf{ai}{\uparrow}, \mathsf{q}{\uparrow}, \mathsf{u}{\uparrow}\}$ is admissible for BVB one independently from the other.

We start proving three lemmas which all are consequences of splitting. They say that the up rules of SBVB are admissible for BVB whenever we apply them in a shallow context $[\{\ \} \parr P]$. Context reduction will extend the three lemmas to any context.

LEMMA 3.6 ($\mathsf{ai}{\uparrow}$ ***is admissible for*** BVB ***in shallow contexts***)
Let P be a structure and let $\mathfrak{n}$ be a name. If $\Phi : \vdash_{\mathsf{BVB}} [(\mathfrak{n} \otimes \overline{\mathfrak{n}}) \parr P]$ then $\Psi : \vdash_{\mathsf{BVB}} P$.

PROOF. This proof is identical to the one in [8]. Splitting for Seq and CoPar (Lemma 3.1) applies to Φ. There are:

$$\begin{array}{c} [P_1 \parr P_2] \\ \Gamma \Big\Vert \mathsf{BVB} \\ P \end{array} \quad \text{and} \quad \begin{array}{c} \Phi_1 \\ \Big\Vert \mathsf{BVB} \\ [\mathfrak{n} \parr P_1] \end{array} \quad \text{and} \quad \begin{array}{c} \Phi_2 \\ \Big\Vert \mathsf{BVB} \\ [\overline{\mathfrak{n}} \parr P_2] \end{array}.$$

Splitting for names (Lemma 3.3) applies to Φ_1 and Φ_2 to yield $\Delta_1 : \overline{\mathfrak{n}} \vdash_{\mathsf{BVB}} P_1$ and $\Delta_2 : \mathfrak{n} \vdash_{\mathsf{BVB}} P_2$. We can conclude by applying an instance of $\mathsf{ai}{\downarrow}$ followed by Δ_1, Δ_2 and Γ. ■

LEMMA 3.7 ($\mathsf{q}{\uparrow}$ ***is admissible for*** BVB ***in shallow contexts***)
Let R, T, U, V and P be structures. If $\Phi : \vdash_{\mathsf{BVB}} [(\langle R \lhd U \rangle \otimes \langle T \lhd V \rangle) \parr P]$ then $\Psi : \vdash_{\mathsf{BVB}} [\langle (R \otimes T) \lhd (U \otimes V) \rangle \parr P]$.

PROOF. This proof is identical to the one in [8]. We can repeatedly apply Splitting for Seq and CoPar (Lemma 3.1), starting from Φ. There are:

$$\begin{array}{c} [\langle P_1 \lhd P_2 \rangle \parr \langle P_3 \lhd P_4 \rangle] \\ \Gamma \Big\Vert \mathsf{BVB} \\ P \end{array}, \quad \begin{array}{c} \Phi_1 \\ \Big\Vert \mathsf{BVB} \\ [R \parr P_1] \end{array}, \quad \begin{array}{c} \Phi_2 \\ \Big\Vert \mathsf{BVB} \\ [T \parr P_2] \end{array}, \quad \begin{array}{c} \Phi_3 \\ \Big\Vert \mathsf{BVB} \\ [U \parr P_3] \end{array} \quad \text{and} \quad \begin{array}{c} \Phi_4 \\ \Big\Vert \mathsf{BVB} \\ [V \parr P_4] \end{array}.$$

Let us compose Φ_1, Φ_2, Φ_3 and Φ_4 for proving $\langle([R \parr P_1] \otimes [T \parr P_2]) \triangleleft ([U \parr P_3] \otimes [V \parr P_4])\rangle$. We conclude by applying instances of s and $\mathsf{q}{\downarrow}$ followed by Γ. ■

LEMMA 3.8 ($\mathsf{u}{\uparrow}$ is admissible for BVB in shallow contexts)
Let R, T and P be structures. If $\Phi : \vdash_{\mathsf{BVB}} [(\lceil R \rfloor_a \otimes \lceil T \rfloor_a) \parr P]$ then $\Psi : \vdash_{\mathsf{BVB}} [\lceil (R \otimes T) \rfloor_a \parr P]$.

PROOF. Splitting for Seq and CoPar (Lemma 3.1) applies to Φ. There are:

$$\begin{array}{c}[P_1 \parr P_2] \\ \Gamma \,\|_{\mathsf{BVB}} \\ P\end{array} \quad , \quad \begin{array}{c}\Phi_1 \,\|_{\mathsf{BVB}} \\ [\lceil R \rfloor_a \parr P_1]\end{array} \quad \text{and} \quad \begin{array}{c}\Phi_2 \,\|_{\mathsf{BVB}} \\ [\lceil T \rfloor_a \parr P_2]\end{array} .$$

Splitting for Sdb (Lemma 3.2) applies to Φ_1 and Φ_2. There are:

$$\begin{array}{c}\lceil U_1 \rfloor_a \\ \Delta_1 \,\|_{\mathsf{BVB}} \\ P_1\end{array} \quad , \quad \begin{array}{c}\Psi_1 \,\|_{\mathsf{BVB}} \\ [R \parr U_1]\end{array} \quad , \quad \begin{array}{c}\lceil U_2 \rfloor_a \\ \Delta_2 \,\|_{\mathsf{BVB}} \\ P_2\end{array} \quad \text{and} \quad \begin{array}{c}\Psi_2 \,\|_{\mathsf{BVB}} \\ [T \parr U_2]\end{array} .$$

We can conclude as follows:

$$\begin{array}{c}\Psi_1 \,\|_{\mathsf{BVB}} \\ \lceil ([R \parr U_1] \otimes \circ) \rfloor_a \\ \Psi_2 \,\|_{\mathsf{BVB}} \\ \{\mathsf{s}^2, (11)^2\} \dfrac{\lceil ([R \parr U_1] \otimes [T \parr U_2]) \rfloor_a}{\lceil [(R \otimes T) \parr U_1 \parr U_2] \rfloor_a} \\ \{\mathsf{u}{\downarrow}^2\} \dfrac{}{[\lceil (R \otimes T) \rfloor_a \parr \lceil U_1 \rfloor_a \parr \lceil U_2 \rfloor_a} \\ \Delta_1 \,\|_{\mathsf{BVB}} \\ [\lceil (R \otimes T) \rfloor_a \parr P_1 \parr \lceil U_2 \rfloor_a] \\ \Delta_2 \,\|_{\mathsf{BVB}} \\ [\lceil (R \otimes T) \rfloor_a \parr P_1 \parr P_2] \\ \Gamma \,\|_{\mathsf{BVB}} \\ [\lceil (R \otimes T) \rfloor_a \parr P]\end{array} .$$

■

PROPOSITION 3.9 (Up-rules are admissible for BVB)
Let R, T, U, and V, be structures and $S\{\ \}$ be a context.

(1) If $\Phi : \vdash_{\mathsf{BVB}} S(a \otimes \overline{a})$ then there exists $\Psi : \vdash_{\mathsf{BVB}} S\{\circ\}$.
(2) If $\Phi : \vdash_{\mathsf{BVB}} S(\langle R \triangleleft U\rangle \otimes \langle T \triangleleft V\rangle)$ then there exists $\Psi : \vdash_{\mathsf{BVB}} S\langle (R \otimes T) \triangleleft (U \otimes V)\rangle$.
(3) If $\Phi : \vdash_{\mathsf{BVB}} S(\lceil R \rfloor_a \otimes \lceil T \rfloor_a)$ then there exists $\Psi : \vdash_{\mathsf{BVB}} S\lceil (R \otimes T) \rfloor_a$.

PROOF. The proofs of the three points proceeds similarly one another. We recall the proof of the first point from [8] and detail the proof of the third one which is specific to BVB.

Point 1. Context reduction (Proposition 3.4) applies to the assumption $\Phi : \vdash_{\mathsf{BVB}} S(a \otimes \overline{a})$. There are a structure U and a, possibly empty, sequence of variables $\vec{b}$ such that, for every V which is bound-variable equivalent with $(a \otimes \overline{a})$, we have $\Gamma : \lceil [V \parr U] \rfloor_{\vec{b}} \vdash_{\mathsf{BVB}} S\{V\}$ and

$\Psi: \vdash_{\mathsf{BVB}} [(a \otimes \overline{a}) \parr U]$. In particular, it exists $\Delta: \lceil [\circ \parr U] \rfloor_{\vec{b}} \vdash_{\mathsf{BVB}} S\{\circ\}$. Lemma 3.6 applies to Ψ saying that $\mathsf{ai}{\uparrow}$ is admissible for BVB in shallow contexts, i.e. $\Psi': \vdash_{\mathsf{BVB}} U$ and, consequently, $\Psi'': \vdash_{\mathsf{BVB}} \lceil [\circ \parr U] \rfloor_{\vec{b}}$. We can conclude by concatenating Ψ'' and Δ.

Point 3. Context reduction (Proposition 3.4) applies to $\Phi: \vdash_{\mathsf{BVB}} S(\lceil R \rfloor_a \otimes \lceil T \rfloor_a)$. There are a structure U and a, possibly empty, sequence of variables $\vec{b}$ such that, for every V which is bound-variable equivalent with $(\lceil R \rfloor_a \otimes \lceil T \rfloor_a)$, we have $\Gamma: \lceil [V \parr U] \rfloor_{\vec{b}} \vdash_{\mathsf{BVB}} S\{V\}$ and $\Psi: \vdash_{\mathsf{BVB}} [(\lceil R \rfloor_a \otimes \lceil T \rfloor_a) \parr U]$. In particular, it exists $\Delta: \lceil [\lceil (R \otimes T) \rfloor_a \parr U] \rfloor_{\vec{b}} \vdash_{\mathsf{BVB}} S \lceil (R \otimes T) \rfloor_a$. Lemma 3.8 applies to Ψ saying that $\mathsf{u}{\uparrow}$ is admissible for BVB in shallow contexts, i.e. $\Psi': \vdash_{\mathsf{BVB}} [\lceil (R \otimes T) \rfloor_a \parr U]$ and, consequently, $\Psi'': \vdash_{\mathsf{BVB}} \lceil [\lceil (R \otimes T) \rfloor_a \parr U] \rfloor_{\vec{b}}$. We can conclude by concatenating Ψ'' and Δ. ■

THEOREM 3.10 (The Up fragment is admissibile for BVB)
It is possible to remove every instance of the rules $\{\mathsf{ai}{\uparrow}, \mathsf{q}{\uparrow}, \mathsf{u}{\uparrow}\}$ from any proof $\Phi: \vdash_{\mathsf{SBVB}} R$.

PROOF. By iteratively applying Proposition 3.9, starting from Φ. ■

Theorem 3.10 here above directly implies:

COROLLARY 3.11 (The cut-elimination holds for SBVB)
For every proof $\Phi: \vdash_{\mathsf{SBVB}} R$ there exists $\Psi: \vdash_{\mathsf{BVB}} R$ with no instances of $\mathsf{ai}{\uparrow}$.

4 Linear λ-calculus mapped to BVB

We give a semantic interpretation to Sdb. Its ability to bind names and its interplay with Seq can model linear β-reduction of linear λ-calculus.

Linear λ-calculus: It is a restriction of the standard λ-calculus whose syntax we recall in (27):

$$M ::= \mathcal{V} \mid \lambda\mathcal{V}.M \mid (M)M \tag{27}$$

The symbol $\mathcal{V}$ in (27) denotes the countable set of variable names we range over by x, y, w, z. Every $\lambda x.M$ is an abstraction and every $(M)M$ is an application.

To define linear λ-calculus we need to define every linear λ-term together with the set of its free variables to constrain every variable not to occur more than once in the term.

By definition, the set of linear λ-terms is $\Lambda = \bigcup_{X \subset \mathcal{V}} \Lambda_X$ which we range over by M, N, F, G and such that Λ_X contains the *linear λ-terms whose free variables are in X*. For any X, the definition of Λ_X is in (28).

$$\begin{array}{l} x \in \Lambda_{\{x\}} \\ \lambda x.M \in \Lambda_X \text{ if } M \in \Lambda_{X \cup \{x\}} \\ (M)N \in \Lambda_{X \cup Y} \text{ if } M \in \Lambda_X,\ N \in \Lambda_Y \text{ and } X \cap Y = \emptyset \end{array} \tag{28}$$

EXAMPLE 4.1
Neither $\lambda x.\lambda y.x$ nor $\lambda x.(x)x$ are linear. They violates the second and the third clause, respectively. Instead $\lambda z.\lambda x.\lambda y.((z)y)x$, $\lambda w.\lambda z.(w)z$ and $\lambda w.\lambda z.(z)w$ belong to linear λ-calculus.

The operational semantics that rewrites linear λ-terms is the *linear β-reduction* $\Rightarrow \subseteq \Lambda \times \Lambda$ in (29).

$$\mathsf{rfl}\frac{}{M \Rightarrow M} \qquad \beta\frac{}{(\lambda x.M)N \Rightarrow (M)\{x:=N\}} \qquad \mathsf{tra}\frac{M \Rightarrow F \qquad F \Rightarrow N}{M \Rightarrow N}$$
$$\mathsf{f}\frac{M \Rightarrow N}{\lambda x.M \Rightarrow \lambda x.N} \qquad @\mathsf{l}\frac{M \Rightarrow N}{(M)F \Rightarrow (N)F} \qquad @\mathsf{r}\frac{M \Rightarrow N}{(F)M \Rightarrow (F)N} \tag{29}$$

In (29), $(M)\{N:=x\}$ is the usual meta operation *clash-free substitution*. It replaces N for the forcefully single occurrence of x in M. Finally, in analogy to the length of a derivation in SBVB, $\|M \Rightarrow N\|$ denotes the *number of instances of rules* in (29) used to derive a given $M \Rightarrow N$.

The map $(\!|\_|\!)\_$: The three clauses (30a), (30b) and (30c) define it by mapping terms of Λ to structures of BVB.

$$(\!|x|\!)_o = \langle x \lhd \overline{o} \rangle \tag{30a}$$
$$(\!|\lambda x.M|\!)_o = \lceil (\!|M|\!)_o \rfloor_x \tag{30b}$$
$$(\!|(M)N|\!)_o = \lceil [(\!|M|\!)_p \parr \lceil (\!|N|\!)_q \rfloor_q \parr \langle p \lhd \overline{o} \rangle] \rfloor_p \text{ with } p,q \text{ fresh} \tag{30c}$$

We think that reading the embedding $(\!|\_|\!)\_$ as it was a map from λ-terms to processes is the best way to have a catch on why $(\!|\_|\!)\_$ works so that we can prove the completeness of proof-search in BVB w.r.t. linear λ-calculus. The map $(\!|\_|\!)\_$ allows to reconstruct communication paths inside a linear λ-term. Every path links an input to the output. The reconstruction of every path relies on merging names of input channels with names of output channels and on preserving the order relation associated to Seq. A little bit more technically, we can describe how $(\!|\_|\!)\_$ works by rephrasing the description of the embedding of the λ-calculus to π-calculus in [22], source of inspiration for $(\!|\_|\!)\_$.

- The clause (30a) sees the variable x as an input channel which retransmits what it takes in input to the output channel o.
- For any abstraction $\lambda x.M$, the clause (30b) gives the name o to the output of M which also becomes the output name of the whole $\lambda x.M$. We bind the input x of M by means of Sdb letting it ready to merge with suitably bound output channels. We carry out merging by means of the renaming which is part of the definition of Sdb.
- For every application $(M)N$, we bind the output of the interpretation of N which becomes ready to be merged with the input channel of $(\!|M|\!)_p$ in case it evaluates to the interpretation of some abstraction. The outermost instance of Sdb hides the output channel of $(\!|M|\!)_p$.

5 Completeness of BVB w.r.t. Linear λ-calculus

Completeness says that we can mimic every computation step of linear λ-calculus as proof-reconstruction inside BVB. The derivation of BVB that simulates one linear β-reduction step for any λ-terms M and N is in (31).

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(\!|(M)\{x{:=}N\}|\!)_o}{\ulcorner(\!|(M)\{x{:=}N\}|\!)_o\urcorner_p}\{(18)\}}{\ulcorner[(\!|(M)\{x{:=}N\}|\!)_p \parr \langle p \lhd \overline{o}\rangle]\urcorner_p}\mathsf{mt}{\downarrow}}{\ulcorner[\ulcorner(\!|(M)\{x{:=}N\}|\!)_p\urcorner_x \parr \langle p \lhd \overline{o}\rangle]\urcorner_p}\{(18)\}}{\ulcorner[\ulcorner[\ulcorner(\!|M|\!)_p \parr (\!|N|\!)_x]\urcorner_x \parr \langle p \lhd \overline{o}\rangle]\urcorner_p}\mathsf{subst}}{\ulcorner[\ulcorner[\ulcorner(\!|M|\!)_p\urcorner_x \parr \ulcorner(\!|N|\!)_x\urcorner_x \parr \langle p \lhd \overline{o}\rangle]\urcorner_p}\mathsf{u}{\downarrow}}{\ulcorner[\ulcorner[\ulcorner(\!|M|\!)_p\urcorner_x \parr \ulcorner(\!|N|\!)_q\urcorner_q \parr \langle p \lhd \overline{o}\rangle]\urcorner_p}\{(19)\} \tag{31}$$

We analyse (31) starting from its conclusion.

From (30a), (30b) and (30c) we know that $\ulcorner[\ulcorner(\!|M|\!)_p\urcorner_x \parr \ulcorner(\!|N|\!)_q\urcorner_q \parr \langle p \lhd o\rangle]\urcorner_p$ is equal to $(\!|(\lambda x.M)N|\!)_o$. We can apply (19) for $\ulcorner(\!|N|\!)_q\urcorner_q \approx \ulcorner(\!|N|\!)_q\{^x/_q\}\urcorner_x$ because input/output channels are unique. The instance of $\mathsf{u}{\downarrow}$ identifies the input channel x of $\ulcorner(\!|M|\!)_p\urcorner_x$ and the output channel x of $(\!|N|\!)_x$. The definition of $\mathsf{mt}{\downarrow}$ and subst are in (32).

$$\mathsf{mt}{\downarrow}\dfrac{(\!|M|\!)_o}{[(\!|M|\!)_p \parr \langle p \lhd \overline{o}\rangle]} \qquad \mathsf{subst}\dfrac{(\!|(M)\{x{:=}N\}|\!)_o}{[(\!|M|\!)_o \parr (\!|N|\!)_x]} \tag{32}$$

We will show how to derive $\mathsf{mt}{\downarrow}$ and subst in BVB in a few. The instance of the rule subst in (31) occurs deeply in its context. It mimics the substitution on linear λ-terms. The first occurrence of (18) applies because x disappears. The topmost occurrence of (18) applies because p disappears as consequence of the application of $\mathsf{mt}{\downarrow}$ which relies on Seq-transitivity in (26).

5.1 The proof of completeness

LEMMA 5.1 (Output names are linear)
For every linear λ-term M, the variable name o of $(\!|M|\!)_o$ occurs once.

PROOF. By induction on the definition of $(\!|\_|\!)_\_$, proceeding by case analysis on the form of M. ■

LEMMA 5.2 ($\mathsf{mt}{\downarrow}$ is derivable in BVB)
Let M and N be linear λ-terms. For every variable names p and o, we can derive the rule $\mathsf{mt}{\downarrow}$ of (32) in BVB.

PROOF. We reason by induction on the size $|(\!|M|\!)_o|$, of $(\!|M|\!)_o$ in the conclusion of $\mathsf{mt}{\downarrow}$, proceeding by case analysis on the form of M.

1. A first base case is with $M \equiv x$. By definition, $(\!|x|\!)_p$ is $\langle x \lhd \overline{p}\rangle$. Then:

$$\mathsf{t}{\downarrow}\dfrac{\langle x \lhd \overline{o}\rangle}{[\langle x \parr \overline{p}\rangle \parr \langle p \lhd \overline{o}\rangle]}\ .$$

   The premise is equivalent to $(\!|x|\!)_o$.

2. The second base case is with $M \equiv (M')M''$. By definition $(\!|(M')M''|\!)_p$ is equal to $\lceil[(\!|M'|\!)_r \parr \lceil(\!|M''|\!)_q\rfloor_q \parr \langle r \lhd \overline{p}\rangle]\rfloor_r$, for some r. Then:

$$\{(18),\mathsf{u}{\downarrow}\}\dfrac{\mathsf{t}{\downarrow}\dfrac{\lceil[(\!|M'|\!)_r \parr \lceil(\!|M''|\!)_q\rfloor_q \parr \langle r \lhd \overline{o}\rangle]\rfloor_r}{\lceil[(\!|M'|\!)_r \parr \lceil(\!|M''|\!)_q\rfloor_q \parr \langle r \lhd \overline{p}\rangle \parr \langle p \lhd \overline{o}\rangle]\rfloor_r}}{[\lceil[(\!|M'|\!)_r \parr \lceil(\!|M''|\!)_q\rfloor_q \parr \langle r \lhd \overline{p}\rangle]\rfloor_r \parr \langle p \parr \overline{o}\rangle]} \ .$$

   The premise is equivalent to $(\!|(M')M''|\!)_o$.
3. The unique inductive case is with $M \equiv \lambda y.M'$. By definition, $(\!|\lambda y.M'|\!)_p$ is $\lceil(\!|M'|\!)_p\rfloor_y$. Then:

$$\{(18),\mathsf{u}{\downarrow}\}\dfrac{\mathsf{mt}{\downarrow}\dfrac{\lceil(\!|M'|\!)_o\rfloor_y}{\lceil[(\!|M'|\!)_p \parr \langle p \lhd \overline{o}\rangle]\rfloor_y}}{[\lceil(\!|M'|\!)_p\rfloor_y \parr \langle p \lhd \overline{o}\rangle]} \ ,$$

   where $\mathsf{mt}{\downarrow}$ applies by induction because $|(\!|M'|\!)_p| < |(\!|\lambda y.M'|\!)_p|$. The premise is equivalent to $(\!|\lambda y.M'|\!)_o$. ■

LEMMA 5.3 (subst is derivable in BVB)
Let M and N be linear λ-terms. For every variable names o and x such that $x \in \mathrm{fn}((\!|M|\!)_o)$, we can derive the rule subst of (32) in BVB.

PROOF. We reason by induction on the size $|[(\!|M|\!)_o \parr (\!|N|\!)_x]|$, proceeding by case analysis on the form of M and N.

1. Let $M \equiv x$. We have three sub cases, depending on N.
   (a) Let $N \equiv y$. By definition, $[(\!|x|\!)_o \parr (\!|y|\!)_x]$ is $[\langle x \lhd \overline{o}\rangle \parr \langle y \lhd \overline{x}\rangle]$. Then:

$$\mathsf{t}{\downarrow}\dfrac{\langle y \lhd \overline{o}\rangle}{[\langle x \lhd \overline{o}\rangle \parr \langle y \lhd \overline{x}\rangle]} \ .$$

   The premise is $(\!|(x)\{x{:=}y\}|\!)_o \approx (\!|y|\!)_o$.
   (b) Let $N \equiv (N')N''$. By definition, $[\langle x \lhd \overline{o}\rangle \parr \lceil[(\!|N'|\!)_p \parr \lceil(\!|N''|\!)_q\rfloor_q \parr \langle p \lhd \overline{x}\rangle]\rfloor_p]$ is $[(\!|x|\!)_o \parr (\!|(N')N''|\!)_x]$. Then:

$$\{(14),\mathsf{u}{\downarrow},(18)\},(10)\dfrac{\mathsf{t}{\downarrow}\dfrac{\lceil[(\!|N'|\!)_p \parr \lceil(\!|N''|\!)_q\rfloor_q \parr \langle p \lhd \overline{o}\rangle]\rfloor_p}{\lceil[(\!|N'|\!)_p \parr \lceil(\!|N''|\!)_q\rfloor_q \parr \langle p \lhd \overline{x}\rangle \parr \langle x \lhd \overline{o}\rangle]\rfloor_p}}{[\langle x \lhd \overline{o}\rangle \parr \lceil[(\!|N'|\!)_p \parr \lceil(\!|N''|\!)_q\rfloor_q \parr \langle p \lhd \overline{x}\rangle]\rfloor_p]} \ .$$

   The premise is $(\!|(x)\{x{:=}(N')N''\}|\!)_o \approx (\!|(N')N''|\!)_o$.
   (c) Let $N \equiv \lambda y.N'$. Without loss of generality, we can assume $y \not\equiv x$. By definition $[(\!|x|\!)_o \parr (\!|\lambda y.N'|\!)_x] \approx [\langle x \lhd \overline{o}\rangle \parr \lceil(\!|N'|\!)_x\rfloor_y]$. Then:

$$\{(18),\mathsf{u}{\downarrow}\}\dfrac{\mathsf{mt}{\downarrow}\dfrac{\lceil(\!|N'|\!)_o\rfloor_y}{\lceil[\langle x \lhd \overline{o}\rangle \parr (\!|N'|\!)_x]\rfloor_y}}{[\langle x \lhd \overline{o}\rangle \parr \lceil(\!|N'|\!)_x\rfloor_y]} \ .$$

   The premise is $(\!|(x)\{x{:=}\lambda y.N'\}|\!)_o \approx (\!|\lambda y.N'|\!)_o$.

2. Let $M \equiv \lambda y.M'$. Without loss of generality we assume $y \not\equiv x$. By definition, $(\!|\lambda y.M'|\!)_o$ is $\lceil (\!|M'|\!)_o \rfloor_y$. Then:

$$\mathsf{subst}\,\{(18),\mathsf{u}{\downarrow}\}\frac{\lceil (\!|(M')\{x:=N'\}|\!)_o \rfloor_y}{\dfrac{\lceil [(\!|M'|\!)_o \parr (\!|N'|\!)_x] \rfloor_y}{[\lceil (\!|M'|\!)_o \rfloor_y \parr (\!|N|\!)_x]}}\ ,$$

where subst applies by induction because $|[(\!|M'|\!)_o \parr (\!|N'|\!)_x]| < |[(\!|\lambda y.M'|\!)_o \parr (\!|N'|\!)_x]|$. The premise is equivalent to $(\!|\lambda y.(M')\{x:=N'\}|\!)_o$.

3. Let $M \equiv (M')M''$ with $x \in \mathrm{fv}(M')$. By definition, $\lceil [(\!|M'|\!)_p \parr \lceil (\!|M''|\!)_q \rfloor_q \parr \langle p \triangleleft \overline{o} \rangle] \rfloor_p$ is $(\!|(M')M''|\!)_o$. Then:

$$\mathsf{subst}\,\{(18),\mathsf{u}{\downarrow},(14),(10)\}\frac{\lceil [(\!|(M')\{x:=N\}|\!)_p \parr \lceil (\!|M''|\!)_q \rfloor_q \parr \langle p \triangleleft \overline{o} \rangle] \rfloor_p}{[\lceil [(\!|M'|\!)_p \parr (\!|N|\!)_x] \triangleleft \lceil (\!|M''|\!)_q \rfloor_q \parr \langle p \triangleleft \overline{o} \rangle] \rfloor_p}\ ,$$

where $|[(\!|M'|\!)_p \parr (\!|N|\!)_x]| < |[(\!|(M')M''|\!)_o \parr (\!|N|\!)_x]|$ allows to apply subst by the inductive argument. The premise is equivalent to $(\!|((M')\{x:=N\})M''|\!)_o$.

4. Let $M \equiv (M')M''$ with $x \in \mathrm{fv}(M'')$. By definition, $\lceil [(\!|M'|\!)_p \parr \lceil (\!|M''|\!)_q \rfloor_q \parr \langle p \triangleleft \overline{o} \rangle] \rfloor_p$ is $(\!|(M')M''|\!)_o$. Then:

$$\mathsf{subst}\,\{(18),\mathsf{u}{\downarrow},(14),(10)\}\frac{\lceil [(\!|M'|\!)_p \parr \lceil (\!|(M'')\{x:=N\}|\!)_q \rfloor_q \parr \langle p \triangleleft \overline{o} \rangle] \rfloor_p}{[\lceil [(\!|M'|\!)_p \parr \lceil (\!|M''|\!)_q \rfloor_q \parr (\!|N|\!)_x] \parr \langle p \triangleleft \overline{o} \rangle] \rfloor_p \parr (\!|N|\!)_x]}\ ,$$

where $|[(\!|M''|\!)_q \parr (\!|N|\!)_x]| < |[(\!|(M')M''|\!)_o \parr (\!|N|\!)_x]|$ allows to apply subst by the inductive argument. The premise is $(\!|(M')(M''\{x:=N\})|\!)_o$. ■

LEMMA 5.4 (Linear β-reduction in BVB)
Let M and N be λ-terms. Let o and x be variable names. We can derive the following rule in BVB:

$$\mathsf{beta}\frac{(\!|(M)\{x:=N\}|\!)_o}{(\!|(\lambda x.M)N|\!)_o}\ .$$

PROOF. The derivation of beta is in (31). It relies on the definition of $(\!|\_|\!)_\_$ and on Lemma 5.2 and 5.3. ■

THEOREM 5.5 (Completeness of BVB)
Let M and N be linear λ-terms. Let o be a variable name. If $M \Rightarrow N$ then $\Gamma : (\!|N|\!)_o \vdash_{\mathsf{BVB}} (\!|M|\!)_o$, for some derivation Γ.

PROOF. By induction on $\|M \Rightarrow N\|$, proceeding by case analysis on the lowermost rule instance used to derive $M \Rightarrow N$. If the lowermost rule instance is β then Lemma 5.4 implies the thesis. Let

the lowermost rule instance be tra. The inductive hypothesis implies the existence of Γ_0 and Γ_1 such that:

$$\begin{array}{l} (\!|N|\!)_o \\ \Gamma_1 \,\|\, \mathsf{BVB} \\ (\!|F|\!)_o \\ \Gamma_0 \,\|\, \mathsf{BVB} \\ (\!|M|\!)_o \end{array} .$$

In all the remaining cases we proceed analogously, exploiting that BVB is a deep inference system, so we can apply deeply, in any context, any of its rules. ■

REMARK 5.6
As a corollary, under the same assumption as in Theorem 5.5, we have $\vdash_{\mathsf{BVB}} [(\!|M|\!)_o \parr \overline{(\!|N|\!)_o}]$ because we can derive $\mathsf{i}{\downarrow}$ in BVB, and we can plug it on top of Γ. ■

6 Conclusions and future work

As far as the computational interpretation of proof-search inside BVB is concerned, this work makes no reference to soundness of BVB w.r.t. linear λ-calculus, a weak version of which we prove in [16, 17]. Full soundness would say when a derivation in BVB describes a computation of linear λ-calculus, i.e. when, for every M, N and o, if $\begin{array}{l} (\!|N|\!)_o \\ \Gamma \,\|\, \mathsf{BVB} \\ (\!|M|\!)_o \end{array}$ then $M \Rightarrow N$. Ideas from [18], which allow to prove that NEL is undecidable, might help to prove full soundness having linear λ-calculus as target language possibly after some variations of $(\!|\_|\!)\_$. We leave it as possible future work for we plan to pursue the programme that [2] begins. We want to move beyond linear λ-calculus as a target. The preliminary results in [15] suggest that the natural computational paradigm w.r.t. which BVB can be sound is some strict extension of $\mathsf{CC_{sp}}$, i.e. of the fragment of CCS [13] with sequential and parallel composition only.

As far as the proof-theory of BVB is concerned, we aim at the minimal and incremental extension of SBV, an example of which is SBVB, to keep investigating self-dual operators. By means of a self-dual operator and in accordance with the proof-search-as-computation paradigm we plan to model non-deterministic choice. Candidate rules that model a self-dual non-deterministic choice are:[1]

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

We think they are interesting because they would internalize the non deterministic choice that we apply at the meta-level when searching for proofs, or derivations, inside SBVB or SBV.

Acknowledgments

They are due to Paola Bruscoli and Lutz Straßburger who, thanks to their detailed comments, helped to improve the presentation of this work.

[1] The conjecture about the existence of the two rules $\mathsf{p}{\downarrow}, \mathsf{p}{\uparrow}$ able to model non-deterministic choice results from discussions with Alessio Guglielmi.

Funding

Partially supported by the Royal Society project n. IE111499 "Sharing and Sequentiality in Proof Systems with Locality".

References

[1] K. Brünnler and R. McKinley. An algorithmic interpretation of a deep inference system. In *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, Vol. 5330 of *Lecture Notes in Computer Science*, I. Cervesato, H. Veith, and A. Voronkov, eds, pp. 482–496. Springer-Verlag, 2008.

[2] P. Bruscoli. A purely logical account of sequentiality in proof search. In *Logic Programming, 18th International Conference*, Vol. 2401 of *Lecture Notes in Computer Science*, P. J. Stuckey, ed., pp. 302–316. Springer-Verlag, 2002.

[3] J.-Y. Girard, P. Taylor, and Y. Lafont. *Proofs and Types*. Cambridge University Press, 1989.

[4] N. Guenot. *Nested Deduction in Logical Foundations for Computation*. PhD Thesis, Ecole Polytechnique — Laboratoire d'Informatique (LIX), rue de Saclay, 91128 Palaiseau cedex, 2013.

[5] A. Guglielmi. A system of interaction and structure. *ACM Transactions on Computational Logic*, **8**, 1–64, 2007.

[6] A. Guglielmi and L. Straßburger. Non-commutativity and MELL in the calculus of structures. In *CSL 2001*, Vol. 2142 of *Lecture Notes in Computer Science*, L. Fribourg, ed., pp. 54–68. Springer-Verlag, 2001.

[7] A. Guglielmi and L. Straßburger. A non-commutative extension of MELL. In *LPAR 2002*, Vol. 2514 of *Lecture Notes in Computer Science*, M. Baaz and A. Voronkov, eds, pp. 231–246. Springer-Verlag, 2002.

[8] A. Guglielmi and L. Straßburger. A system of interaction and structure V: the exponentials and splitting. *Mathematical Structures in Computer Science*, **21**, 563–584, 2011.

[9] T. Gundersen, W. Heijltjes, and M. Parigot. Atomic lambda calculus: a typed lambda-calculus with explicit sharing. In *LICS*, pp. 311–320. IEEE Computer Society, 2013.

[10] K. Honda and N. Yoshida. On the reduction-based process semantics. *Theoretical Computer Science*, **151**, 437—486, 1995.

[11] H. G. Mairson. Linear lambda calculus and ptime-completeness. *Journal of Functional Programming*, **14**, 623–633, 2004.

[12] D. Miller and A. Tiu. A proof theory for generic judgments. *ACM Transactions on Computational Logic*, **6**, 749–783, 2005.

[13] R. Milner. *Communication and Concurrency*. *International Series in Computer Science*. Prentice Hall, 1989.

[14] C. Retoré. A non-commutative extension of classical linear logic. In *TLCA*, Vol. 1210 of *Lecture Notes in Computer Science*, P. de Groote, ed., pp. 300–318. Springer, 1997.

[15] L. Roversi. Communication, and concurrency with logic-based restriction inside a calculus of structures. *ArXiv e-prints*, December 2012. http://arxiv.org/abs/1212.4669.

[16] L. Roversi. Linear lambda calculus and deep inference. In *TLCA 2011 - 10th Typed Lambda Calculi and Applications, Part of RDP'11*, Vol. 6690 of *ARCoSS/LNCS*, L. Ong, ed., pp. 184–197. Springer, 2011.

[17] L. Roversi. Linear lambda calculus with explicit substitutions as proof-search in Deep Inference. *CoRR*, abs/1011.3668, May 2011. http://arxiv.org/abs/1011.3668.

[18] L. Straßburger. System NEL is undecidable. In *10th Workshop on Logic, Language, Information and Computation (WoLLIC)*, Vol. 84 of *Electronic Notes in Theoretical Computer Science*, R. De Queiroz, E. Pimentel, and L. Figueiredo, eds, Elsevier, 2003.

[19] L. Straßburger. Some observations on the proof theory of second order propositional multiplicative linear logic. In *Typed Lambda Calculi and Applications*, Vol. 5608 of *Lecture Notes in Computer Science*, P.-L. Curien, ed., pp. 309–324. Springer-Verlag, 2009.

[20] L. Straßburger and A. Guglielmi. A system of interaction and structure IV: The exponentials and decomposition. *ACM Transactions of Computational Logic*, **12**, 23, 2011.

[21] A. Tiu. A system of interaction and structure II: the need for deep inference. *Logical Methods in Computer Science*, **2**, 1–24, 2006.

[22] S. van Bakel and M. G. Vigliotti. A logical interpretation of the λ-calculus into the π-calculus, preserving spine reduction and types. In *CONCUR 2009 - Concurrency Theory, 20th International Conference, CONCUR 2009, Bologna, Italy, September 1-4, 2009. Proceedings*, Vol. 5710 of *Lecture Notes in Computer Science*, M. Bravetti and G. Zavattaro, eds, Vol. of pp. 84–98. Springer, 2009.

Received 18 December 2012