On the Concurrent Computational Content of Intermediate Logics[*]

Federico Aschieri
TU Wien, Vienna, Austria

Agata Ciabattoni
TU Wien, Vienna, Austria

Francesco A. Genco
IHPST, Paris 1 Panthéon-Sorbonne, Paris, France

January 31, 2020

Abstract

We provide a proofs-as-concurrent-programs interpretation for a large class of intermediate logics that can be formalized by cut-free hypersequent calculi. Obtained by adding classical disjunctive tautologies to intuitionistic logic, these logics are used to type concurrent λ-calculi by Curry–Howard correspondence; each of the calculi features a specific communication mechanism, enhanced expressive power when compared to the λ-calculus, and implements forms of code mobility. We thus confirm Avron's 1991 thesis that intermediate logics formalizable by hypersequent calculi can serve as basis for concurrent λ-calculi.

Keywords Proofs-as-programs, intermediate logics, concurrency, natural deduction, λ-calculus, hypersequents

1 Introduction

Definable in three lines, the λ-calculus provides an elegant, yet powerful computational theory. Remarkably, the λ-calculus is also deeply connected to logic: every intuitionistic proof is isomorphic to some λ-term and the proved formula can be interpreted as a type. In this paradigm, known as Curry–Howard correspondence [24], evaluation of λ-terms can be seen as making the corresponding proofs more direct. The main consequence of this proofs-as-programs correspondence is the possibility of extracting correct by construction and terminating programs from logical proofs. Extending the original ideas of Curry and Howard, different logics (e.g. classical [22, 33], linear [11], modal [31]) have been related to different notions of computation, see e.g. [39] for an overview. Particularly interesting and challenging is to relate logic to concurrency, since concurrent programs are notoriously difficult to write and analyze. In this context, starting with the pioneering work of [1], the propositions-as-sessions research line (e.g. [11, 38]) introduced computational interpretations of linear logic that relate sequent calculus derivations with processes in π-calculus [30, 35] – the most widespread formalism for modelling concurrent systems.

Although resulting in a successful model of concurrent computation, the connection relating linear logic and π-calculus [11, 38, 37, 12] is unsuitable as tool for accomplishing our paper's aim. Indeed, not only we are interested in a different notion of computation – *functional* – but we also

[*]Work funded by the FWF Projects: P32080-N31, START Y544-N23 and W1255-N23.

want to unveil the computational content of different logics – *intermediate logics*, i.e. logics lying between classical and intuitionistic logic. The idea that there might be a connection between concurrency and intermediate logics dates back to 1991. In this year Avron introduced [5] a proof calculus based on multisets of sequents (*hypersequents*) for Gödel logic, one of the best known intermediate logics, and speculated that there might be a strong connection between hypersequents and concurrent computation. He suggested that a hypersequent might be seen as the parallel composition of several sequents that can "communicate" thanks to special inference rules. He thus envisaged the possibility of using the intermediate logics that can be captured by cut-free hypersequent calculi as bases for parallel λ-calculi.

Despite their built-in "concurrency", the structure of hypersequent derivations is not isomorphic to a parallel composition of functional programs. This seems to hinder a functional Curry–Howard interpretations of hypersequent calculi, and notwithstanding various attempts (e.g. [23, 8, 9]), Avron's claim has remained unconfirmed for more than 25 years. What we need instead are well-designed natural deduction inferences that can be interpreted as program instructions, and in particular as typed λ-terms. Normalization [34], which corresponds to the execution of the resulting programs, can then be used to obtain analytic proofs, i.e. proofs containing only formulas that are subformulas of hypotheses or conclusion. For the *particular cases* of Gödel-Dummett logic (intuitionistic logic **IL** extended with $\mathsf{Lin} = (A \to B) \vee (B \to A)$) and classical logic (**IL** extended with $\mathsf{EM} = A \vee \neg A$) well-designed calculi were introduced in [2, 3] by extending the natural deduction system NJ for **IL** by rules simulating the hypersequent rules for Lin and EM, respectively. The reduction rules required to obtain analytic proofs led to the definition of the typed concurrent functional languages λ_{G} and λ_{CL}, featuring channels connecting pair of processes only, thus leaving out multi-party communication.

The infinitely many intermediate logics that admit analytic hypersequent calculi have been characterized in [14]. The logics axiomatized by the disjunctive tautologies of [16], which include the axioms Lin and EM, are among them. In particular, [16] interprets disjunctive tautologies as synchronization schemata and provides the corresponding realizers in a concurrent extension of the λ-calculus [22], still however typed by classical logic. The question of developing natural deduction calculi and concurrent Curry–Howard correspondences for the intermediate logics axiomatized by disjunctive tautologies was instead left open.

Curry–Howard Correspondence: the Concurrent Calculi λ_{L}

In this paper we exploit the intuitions in [5, 16] to provide a Curry–Howard correspondence and a computational interpretation for a large class of intermediate logics with analytic hypersequent calculi. We therefore confirm, in a rather general setting, Avron's thesis on the relationship between concurrent functional computation and those logics. In particular, we present suitable natural deduction calculi for them and prove that the normalization process leading to analytic proofs features all the essential elements of concurrent functional programming: λ-terms, parallelism, communication, private channels and process migration. For this purpose, we introduce a computational syntax for natural deduction proofs that will exhibit all the characters of concurrency.

Each of the logics that we consider is defined by extending **IL** by any arbitrary set L of disjunctive axiom schemata $\mathcal{A}$ of the form

$$(F_1 \to G_1) \vee \ldots \vee (F_m \to G_m)$$

such that the list $F_1 \ldots, F_m$ can contain each propositional variable at most once and $\top$ several times, there is an i such that $F_i \neq \top$, and for each i such that $F_i \neq \top$ there is a j such that $F_i = G_j$. Fixing a particular L, the calculus λ_{L} corresponds to the intermediate logic $\mathbf{IL} + \mathrm{L}$.

The class of logics that we consider[1] includes classical logic, cyclic **IL** [28], Gödel logic, and n-valued Gödel logic, with $n \geq 2$.

The natural deduction calculus used as type system for λ_{L} extends NJ by the natural deduction counterpart [15] of the hypersequent rules for $\mathcal{A}$. The decorated version of these rules lead to λ-calculi with parallel operators that bind two or more processes, according to the rule shape, via a variable a, which represents a private communication channel that behaves similarly to the π-calculus restriction operator ν. For instance, the rules for EM and $C_3 = (A \to B) \vee (B \to C) \vee (C \to A)$ are:

$$\dfrac{\begin{array}{cc} \dfrac{u:A}{a^{\neg A}\, u:\bot} & [a^{A}:A] \\ \vdots & \vdots \\ t:F & s:F \end{array}}{s \parallel_a t : F} \qquad\qquad \dfrac{\begin{array}{ccc} \dfrac{u:A}{a^{A\to B}\, u:B} & \dfrac{v:B}{a^{B\to C}\, v:C} & \dfrac{w:C}{a^{C\to A}\, w:A} \\ \vdots & \vdots & \vdots \\ r:F & s:F & t:F \end{array}}{{}_{a}(\, r \parallel s \parallel t) : F}$$

Our natural deduction calculi are defined in a modular way and the rules added to NJ are *higher-level rules*, as defined by Schroeder-Heister [36], since they also discharge rule applications rather than only formulas. We provide a normalization proof that uniformly applies to all the associated λ-calculi and yields terms satisfying the subformula property. The additional reductions of λ_{L} w.r.t. those of simply typed λ-calculus implement communications between concurrent processes and forms of code mobility. Although the typing rules of λ_{L} are a generalization of those in [2, 3], the activation condition in its reduction rules and the normalization proof are very different. Indeed, though uniform and logically grounded, the technique proposed in [2, 3] of firing only communication redexes that violate the subformula property forces the insertion of dummy types and dummy terms all over the programs, thus making their explicit writing a bit cumbersome. Moreover the normalization proofs in [2, 3] do not allow for a simple treatment of the connective $\vee$, as they require us to add several new typing and reduction rules with no interesting computational meaning at the price of a wild bureaucracy. Hence the reduction rules of λ_{L} and normalization strategy mark a radical departure from [2, 3] Our new reductions are indeed logic independent and are activated when messages are values (cf. Definition 3.2). As a result, programs are easier to write and their evaluation transmits all values.

For any $\mathcal{L} := \mathbf{IL} + \mathrm{L}$ and λ_{L} we show the perfect match between computation steps and proof reductions (Subject Reduction), a terminating reduction strategy (Normalization) for the full logic language, $\vee$ included, and the Subformula Property. The latter is obtained by sophisticated reduction rules (*cross reductions*) which are inspired by hypersequent cut-elimination, generalize those in λ_{G} and λ_{CL}, and address a "fundamental problem in any distributed implementation of a statically-typed, higher-order programming language": how to send open processes and transmit function closures between processes; cf. [19] on Cloud Haskell. The calculus λ_{L} is more expressive than the λ-calculus (see Example 3.5) and, unlike λ_{G} and λ_{CL} [2, 3], can implement multiparty communication. Furthermore, the reduction rules of λ_{L}, which are solely based on the shape of terms and do not rely on their types, might shed light on the long-standing problem of defining a well-behaved untyped concurrent λ-calculus.

Although based on the conference papers [2] and [3], the present work substantially extends and generalizes their results. In particular, we introduce Curry–Howard correspondences for infinitely many intermediate logics, which includes the specific logics considered in [2] and [3]; we define a type-independent reduction system for all the presented calculi, and we prove a general normalization result.

[1] We chose to rule out from our investigation some disjunctive axioms that are characterized in [14] by hypersequent rules: those axioms indeed do not seem to provide further interesting communication mechanisms, while they do greatly complicate notation and normalization proofs.

2 Hypersequents, Natural Deduction and Type System

The intermediate logics we consider are obtained by extending **IL** by any set L of disjunctive axiom schemata $\mathcal{A}$ of the form

$$(F_1 \to G_1) \vee \ldots \vee (F_m \to G_m) \tag{1}$$

where: each F_i is either a propositional variable or $\top$; if $F_i \neq \top$ and $i \neq j$, then $F_i \neq F_j$; if $F_i \neq \top$, then $F_i = G_j$ for some $j \in \{1, \ldots, m\}$; for some $i \in \{1, \ldots, m\}$ $F_i \neq \top$. These include classical logic, cyclic **IL** [28], Gödel logic, and n-valued Gödel logic, with $n \geq 2$. Uniform analytic calculi for these logics are defined [14] in the hypersequent framework, a simple generalization of the sequent framework whose basic objects are "parallel" sequents and whose derivations are communicating parallel sequent derivations.

Definition 2.1 (Hypersequent [4])**.** Hypersequents are multisets of sequents, written as $\Gamma_1 \Rightarrow \Pi_1 \mid \cdots \mid \Gamma_n \Rightarrow \Pi_n$ where, for any $i = 1, \ldots n$, $\Gamma_i \Rightarrow \Pi_i$ is an ordinary sequent, called component.

In this section we present suitable natural deduction calculi for the above logics and introduce corresponding typed concurrent λ calculi. Our natural deduction calculi were defined in [15] by reformulating their hypersequent calculi, which we describe below.

A sequent calculus can be trivially embedded into a hypersequent calculus, by just adding to all sequents a context "$G \mid$", standing for a possibly empty hypersequent. This property is exploited for instance in [25], which defines a hypersequent-based conservative extension of linear logic and uses it as type system for the π-calculus. However, the significance of the hypersequent calculus lies in the larger class of logics for which analytic calculi can be given. This additional expressive power of hypersequent calculi w.r.t. sequent calculi is due to the new definable rules which act simultaneously on several components of one or more hypersequents. These rules, intuitively, allow parallel sequents to communicate.

Henceforth we will denote by L any finite set of axiom schemes of the form (1) above. A cut-free hypersequent calculus for $\mathbf{IL} + L$ is defined by adding to the base hypersequent calculus structural rules equivalent to the provability of the axioms in L. Namely, the result is the hypersequent version of the LJ sequent calculus for **IL**, extended with (ew) and (ec) and structural rules corresponding to the axioms in L; the latter rules allow "exchange of information" between different components. Below we present the rules (ew), (ec) and the general form (r), see [14], of the hypersequent rules corresponding to axioms of the form (1):

$$\frac{G \mid \Gamma \Rightarrow \Pi}{G \mid \Gamma \Rightarrow \Pi \mid \Sigma \Rightarrow \Delta}\ (ew) \qquad \frac{G \mid \Gamma \Rightarrow \Pi \mid \Gamma \Rightarrow \Pi}{G \mid \Gamma \Rightarrow \Pi}\ (ec) \qquad \frac{\{G \mid \Gamma_i, \Gamma'_p \Rightarrow \Pi_i\}_{i \neq p,\ i,p=1,\ldots,m}}{G \mid \Gamma'_1, \Gamma_1 \Rightarrow \Pi_1 \mid \cdots \mid \Gamma'_m, \Gamma_m \Rightarrow \Pi_m}\ (r)$$

with possibly $\Gamma'_j, \{\Gamma_i, \Pi_i\} = \emptyset$. Concrete examples of rules (r) are in Table 1.

The connection between concurrent computation and hypersequent calculi was first noted by Avron. His 1991 thesis states, in particular, that it should be possible to use hypersequent calculi "as bases for parallel λ-calculi" [5]. The most direct way to associate a λ-calculus to a proof calculus is by a Curry–Howard correspondence for a natural deduction calculus. The absence of explicit structural elements makes it possible indeed to use formulae as types of λ-terms, rather than complex objects such as sequents. The natural deduction calculi that we use as typing systems have been extracted from hypersequent calculi. Rather than following the approach in [6, 8, 9], which extends natural deduction by simulating the syntactic structure of hypersequents, we endeavoured to keep our natural deduction calculi as simple as possible. In order to do so while preserving the parallel and concurrent behavior of hypersequents, we exploited the embeddings in [15] between hypersequent derivations and derivations using systems

axiom	*hypersequent rule*	*logic*
$A \vee \neg A$	$\dfrac{G \mid \Gamma, \Sigma \Rightarrow \Delta}{G \mid \Gamma \Rightarrow \mid \Sigma \Rightarrow \Delta}$	**CL**
$(A \to B) \vee (B \to A)$	$\dfrac{G \mid \Gamma_1, \Sigma_2 \Rightarrow \Delta \quad G \mid \Sigma_1, \Gamma_2 \Rightarrow \Theta}{G \mid \Gamma_1, \Gamma_2 \Rightarrow \Delta \mid \Sigma_1, \Sigma_2 \Rightarrow \Theta}$	**GL**
$\bigvee_{i=0}^{k-1} (A_i \to A_{i+1}) \vee (\neg A_k)$	$\dfrac{\{G \mid \Gamma_{j+1}, \Delta_j \Rightarrow \Pi_j\}_{j \in \{0,\dots,k-1\}}}{G \mid \Gamma_0, \Delta_0 \Rightarrow \Pi_0 \mid \dots \mid \Gamma_{k-1}, \Delta_{k-1} \Rightarrow \Pi_{k-1} \mid \Gamma_k \Rightarrow}$	$\mathbf{G}_k$
$\bigvee_{i=1}^{k-1} (A_i \to A_{i+1}) \vee (A_k \to A_1)$	$\dfrac{\{G \mid \Gamma_{j+1}, \Delta_j \Rightarrow \Pi_j\}_{j \in \{1,\dots,k-1\}} \quad G \mid \Gamma_1, \Delta_k \Rightarrow \Pi_k}{G \mid \Gamma_1, \Delta_1 \Rightarrow \Pi_1 \mid \dots \mid \Gamma_k, \Delta_k \Rightarrow \Pi_k}$	$\mathbf{C}_k$

Table 1: Hypersequent rules and corresponding axioms.

of rules of depth 2 (*2-systems*) [32]. A system[2] of rules is a set of (possibly labelled) sequent rules linked together by some variables and by the requirement for the rules of appearing in a certain order in the derivation. The "depth 2" condition limits to two in each branch the number of sequent rules that have to appear in a certain order. The fact that 2-systems essentially correspond to natural deduction rules that can discharge other rule applications (i.e. they are *higher-level rules* according to the terminology in [36]), was already observed in [15], where the question whether the resulting systems are analytic was left open. Before presenting the general form of the rules consider the following example:

Example 2.1. The hypersequent rule for the linearity axiom $\mathsf{Lin} = (A \to B) \vee (B \to A)$ below on the left is translated in [15] into the 2-system at the center. This 2-system corresponds to the natural deduction rule below on the right:

$$\dfrac{G \mid B, \Gamma_1 \Rightarrow \Pi_1 \qquad G \mid A, \Gamma_2 \Rightarrow \Pi_2}{G \mid A, \Gamma_1 \Rightarrow \Pi_1 \mid B, \Gamma_2 \Rightarrow \Pi_2} \qquad \dfrac{\begin{array}{c}\dfrac{B, \Gamma_1 \Rightarrow \Pi_1}{A, \Gamma_1 \Rightarrow \Pi_1}\\ \vdots \\ \Gamma \Rightarrow \Pi\end{array} \quad \begin{array}{c}\dfrac{A, \Gamma_2 \Rightarrow \Pi_2}{B, \Gamma_2 \Rightarrow \Pi_2}\\ \vdots \\ \Gamma \Rightarrow \Pi\end{array}}{\Gamma \Rightarrow \Pi} \qquad \dfrac{\begin{array}{c}\dfrac{A}{B}\\ \vdots \\ C\end{array} \quad \begin{array}{c}\dfrac{B}{A}\\ \vdots \\ C\end{array}}{C}$$

We show below two derivations of Lin:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{B \Rightarrow B \qquad A \Rightarrow A}{A \Rightarrow B \mid B \Rightarrow A}\,(\mathrm{com})}{A \Rightarrow B \mid \Rightarrow B \to A}}{\Rightarrow A \to B \mid \Rightarrow B \to A}}{\Rightarrow A \to B \mid \Rightarrow (A \to B) \vee (B \to A)}}{\Rightarrow (A \to B) \vee (B \to A) \mid \Rightarrow (A \to B) \vee (B \to A)}}{\Rightarrow (A \to B) \vee (B \to A)} \qquad \dfrac{\dfrac{\dfrac{\dfrac{[A]^1}{B}\,*}{A \to B}\,1}{(A \to B) \vee (B \to A)} \quad \dfrac{\dfrac{\dfrac{[B]^2}{A}\,*}{B \to A}\,2}{(A \to B) \vee (B \to A)}}{(A \to B) \vee (B \to A)}\,*$$

where we signal rule application discharge by $*$; on the left, there is a hypersequent derivation, which intutitively consists of two parallel sequent derivations communicating through the rule

[2]The word "system" is used as in linear algebra, where there are systems of equations with common variables, and each equation is meaningful and can be solved only if considered with the other equations of the system.

$$x^A : A \quad \text{for } x \text{ intuitionistic variable} \qquad \frac{u : \bot}{u\,\mathsf{efq}_P : P} \ \text{with } P \text{ atomic, } P \neq \bot \qquad \frac{\begin{array}{c}[x^A : A]\\ \vdots\\ u : B\end{array}}{\lambda x^A u : A \to B} \qquad \frac{t : A \to B \quad u : A}{tu : B}$$

$$\frac{u : A \quad t : B}{\langle u, t\rangle : A \wedge B} \quad \frac{u : A \wedge B}{u\,\pi_0 : A} \quad \frac{u : A \wedge B}{u\,\pi_1 : B} \quad \frac{u : A}{\iota_0(u) : A \vee B} \quad \frac{u : B}{\iota_1(u) : A \vee B} \quad \frac{u : A \vee B \quad \begin{array}{c}[x^A : A]\\ \vdots\\ w_1 : C\end{array} \quad \begin{array}{c}[y^B : B]\\ \vdots\\ w_2 : C\end{array}}{u\,[x^A.w_1, y^B.w_2] : C}$$

$$\frac{u : A}{a^{A\to B}\, u : B} \qquad \frac{\begin{array}{c}[a^{F_1\to G_1} : F_1 \to G_1]\\ \vdots\\ u_1 : B\end{array} \quad \ldots \quad \begin{array}{c}[a^{F_m\to G_m} : F_m \to G_m]\\ \vdots\\ u_m : B\end{array}}{{}_a(u_1 \parallel \ldots \parallel u_m) : B}\ (\mathcal{A}) \qquad \frac{t_1 : A \quad t_2 : A}{t_1 \parallel t_2 : A}\ \mathit{contr}$$

where $(F_1 \to G_1) \vee \ldots \vee (F_m \to G_m)$ is an instance of $\mathcal{A} \in$ L and all a in u_i for $1 \leq i \leq m$ have the form $a^{F_i \to G_i}$

Table 2: Type assignments for λ_{L}.

(*com*); on the right, these two proofs have been split and have become the two branches of the natural deduction rule for Lin.

The general form of our natural deduction rules – coming from a hypersequent rule (r) corresponding to an axiom of the form (1) above via the 2-system rules below left – is as below right

$$\frac{\begin{array}{c}\dfrac{G_1, \Gamma_1 \Rightarrow \Pi_1}{F_1, \Gamma_1 \Rightarrow \Pi_1}\\ \vdots\\ \Gamma \Rightarrow \Pi\end{array} \quad \ldots \quad \begin{array}{c}\dfrac{G_m, \Gamma_m \Rightarrow \Pi_m}{F_m, \Gamma_m \Rightarrow \Pi_m}\\ \vdots\\ \Gamma \Rightarrow \Pi\end{array}}{\Gamma \Rightarrow \Pi} \qquad\qquad \frac{\begin{array}{c}\dfrac{F_1}{G_1}\\ \vdots\\ C\end{array} \quad \ldots \quad \begin{array}{c}\dfrac{F_m}{G_m}\\ \vdots\\ C\end{array}}{C}$$

These rules were considered in [15] (some in [17, 28]) without any normalization proof.

Since the natural deduction rules above are higher-level rules, not all the branches of a derivation containing them are NJ derivations. To avoid this, we use the equivalent rule ($\mathcal{A}$) in Table 2 below. Normalization results were shown in [2, 3] for two *particular instances* of ($\mathcal{A}$): the rules for Lin and EM. The normalization proof for λ_{L} in this paper works instead for all the ($\mathcal{A}$) rules in a uniform way, Gödel and classical logic being very particular cases.

Theorem 2.1 (Soundness and Completeness)**.** *Let* L *be any finite set of axiom schemes of the form (1) above,* $\mathcal{L}$ *be* **IL** + L *and* NL *the natural deduction system obtained by extending* NJ *with all rules* ($\mathcal{A}$) *for* $\mathcal{A} \in$ L*. For any set* Π *of formulas and formula* A*,* $\Pi \vdash_{\mathrm{NL}} A$ *if and only if* $\Pi \vdash_{\mathcal{L}} A$*.*

Proof. ($\Rightarrow$) Applications of ($\mathcal{A}$) can be simulated by $\vee$ eliminations having as major premise an instance of $\mathcal{A}$. ($\Leftarrow$) Any axiom $\mathcal{A}$ can be derived in NJ + ($\mathcal{A}$). ☐

The λ_{L}-Calculus

For a fixed set L of axioms of the form (1) above, the terms of λ_{L} are defined by the type assignment rules in Table 2.

For any formula A, proof terms may contain **intuitionistic variables** $x_0^A, x_1^A, x_2^A, \ldots$ of type A – which are denoted as x^A, y^A, $z^A, \ldots$ – and **channel variables** $a_0^A, a_1^A, a_2^A, \ldots$ of type A – which are denoted as $a^A, b^A, c^A, \ldots$ and represent a communication channel between parallel processes. We assume that the set of channel variables is partitioned into two disjoint classes:

active channels and **inactive channels**. A term ${}_a(u_1 \parallel \dots \parallel u_m)$ is called an **active session**, whenever a is active. The free and bound variables of an intuitionistic proof term are defined as usual. All the occurrences of a which are free in $u_1, \dots, u_m$ are still free in $u_1 \parallel \dots \parallel u_m$ but are bound in ${}_a(u_1 \parallel \dots \parallel u_m)$.

Notice that according to the typing rules, channel variables can only occur as applied variables and not as standalone terms, unlike intuitionistic variables. The notation

$$\begin{array}{c} a^{F_i \to G_i} : F_i \to G_i \\ \vdots \\ u_i : B \end{array}$$

used in the rule $(\mathcal{A})$ denotes a derivation in which $a^{F_i \to G_i}$ occurs as a free channel variable. Proof-theoretically, the square brackets indicate that the assumption is discharged by the rule.

From a computational perspective the rules $(\mathcal{A})$ produce terms of the shape ${}_a(v_1 \parallel \dots \parallel v_m)$ in which the terms $v_1, \dots, v_m$ are in parallel and can communicate with each other through the channel a. Informally, in order to establish a communication channel connecting two terms v_i and v_j, we require that $a^{A_i \to B_i}$ occurs in v_i, $a^{A_j \to B_j}$ occurs in v_j and A_i is equal to B_j. On one hand, the argument w of a channel application $a^{A_i \to B_i}\, w$ will be interpreted as a message of type A_i that must be *transmitted*; on the other hand, the channel application $a^{A_j \to B_j}\, t$ will *receive* messages of type B_j that will replace the whole channel application $a^{A_j \to B_j}\, t$ upon reception. Thus, in general each channel application may send and receive messages.

The rule *contr* corresponds to the hypersequent (ec); in our type system it is logically redundant, but is useful from a computational perspective for representing parallel processes that cannot communicate between themselves.

If $\Gamma = y_1 : A_1, \dots, y_n : A_n$ and all free variables of a proof term $t : A$ are in $y_1, \dots, y_n$, we write $\Gamma \vdash t : A$. From the logical point of view, t represents a natural deduction derivation of A from the hypotheses $A_1, \dots, A_n$ provided that we interpret the applications of the rule below on the left as applications of the rule below on the right:

$$\frac{u : A}{a^{A \to B}\, u : B} \qquad\qquad \frac{A \to B \quad A}{B}$$

If $\parallel$ does not occur in t, t is a *simply typed* λ*-term* representing an intuitionistic deduction. Notice that simply typed λ-terms may contain applied channel variables which are not bound. In these terms, such variables have no associated reduction rule.

Notation. We assume that the connectives $\to$ and $\wedge$ associate to the right. Moreover, as usual, we define $\neg A$ as $A \to \bot$, $\top$ as $\bot \to \bot$, and we adopt the convention that the empty conjunction, namely the expression $A_1 \wedge \dots \wedge A_n$ for $n = 0$, denotes $\top$. By $\langle t_1, t_2, \dots, t_n \rangle$ we denote the term $\langle t_1, \langle t_2, \dots \langle t_{n-1}, t_n \rangle \dots \rangle\rangle$ (which is $\lambda x^\bot\, x^\bot : \top$, abbreviated as $* : \top$, if $n = 0$) and by $\langle t_1, t_2, \dots, t_n \rangle\, \pi_i$, for $i = 0, \dots, n$, the term $\langle t_1, t_2, \dots, t_n \rangle\, \pi_1 \dots \pi_1 \pi_0$ containing the projections that select the $(i+1)$th element of the sequence. For the sake of simplicity, when an operator ${}_a(\dots \parallel \dots)$ is binary, we denote the terms of the form ${}_a(t \parallel s)$ by $t \parallel_a s$.

3 Reduction Rules of λ_{L}

The typing rules of λ_{L} enable us to compose several terms in parallel and to connect them by communication channels. The reduction rules of the calculus will enable us to evaluate the terms and use the communication channels for transmitting messages between them. From a proof-theoretic point of view, these reductions can be seen as simplifications of the proof corresponding to the term. As we will show in Theorem 5.2, these simplifications yield proofs satisfying the subformula property.

The reductions of λ_{L} implementing communications are based on the notion of value. Such notion applies to a message that has to be transmitted because it has terminated its internal computations but will generate a new computation when plugged in a new computational environment. Hence, using all channels that are applied to values amounts to triggering all communications that are going to produce further computation. This is natural from a computational perspective: we trigger the communications that transmit messages with operational content, but we do not start a communication only to send empty objects like variables. Nonetheless, if we want to obtain terms that enjoy the subformula property, we have to consume all channel variables the type of which is too complex with respect to the rest of the term. Therefore, it is not enough to communicate all values. Indeed, the type of a channel occurrence $a\,t$ does not only depend on its argument t but on the whole term ${}_a(\ldots \parallel \ldots \parallel \ldots)$ in which it is bound. Hence, when we reduce the occurrence of a channel because it is applied to a value, we need to make sure that we eliminate all occurrences of the channel. Formally, we exploit the syntactic distinction between active and non-active channels. A communication will only take place through an active channel. If an occurrence of a channel is applied to a value, we replace all the occurrences of the channel by active ones. We thus create an active session. All occurrences of the channel determining the active session transmit their messages.

Remark 3.1. The word *session* is used here as in the literature about *session types*, that is to refer to a "unit of information exchange between two or more communicating agents" [18]. In λ_{L} active sessions are used to force the exhaustion of all possible communications between the involved processes. In the calculi featuring session types, these communications are further described by a type.

In order to define values, we first need to introduce the notion of stack.

Definition 3.1 (Stack)**.** A *stack* is a, possibly empty, sequence $\sigma = \sigma_1\sigma_2\ldots\sigma_n$ where for $1 \leq i \leq n$ either $\sigma_i = t$ for a term t, or $\sigma_i = \pi_j$ with $j \in \{0, 1\}$, or $\sigma_i = [x_1.u_1, x_2.v_2]$, or efq_P for some atom P. We denote the *empty sequence* by ϵ and the stacks of length 1 by ξ. If t is a proof term, we denote by $t\,\sigma$ the term $(((t\,\sigma_1)\,\sigma_2)\ldots\sigma_n)$.

Our notion of value is defined in order to cover all terms that might directly or indirectly trigger new computation if transmitted to another process. Tuples have a special role in the definition of values because they are often used in λ_{L} as containers of messages. For example, cross reductions introduce new tuples – see the terms $\langle \boldsymbol{y_i} \rangle$ in the definition of cross reductions in Table 3. Thus, if we considered all tuples as values with computational content, we would not be able to show that, as the normalization proceeds, communications transmit simpler and simpler values, and thus we would not be able to prove that the normalization terminates.

Definition 3.2 (Value)**.** A *value* is a term of the form $\langle t_1, \ldots, t_n \rangle$, where for some $1 \leq i \leq n$, $t_i = \lambda x\, s$, $t_i = \iota_i(s)$, $t_i = t\,\mathsf{efq}_P$, $t_i = t[x.u, y.v]$ or $t_i = a\sigma$ for an active channel a.

The following notion will be used in the normalization proof, and in Table 3 that contains the reduction rules for λ_{L}-terms

Definition 3.3 (Activable)**.** A channel a is *activable in* u if av occurs in u for some value v.

Definition 3.4 (Simple Context)**.** A *simple context* $\mathcal{C}[\;]$ is a simply typed λ-term with some fixed variable $[\;]$ occurring exactly once. For any proof term u of the same type as $[\;]$, $\mathcal{C}[u]$ denotes the term obtained replacing $[\;]$ with u in $\mathcal{C}[\;]$, *without renaming of bound variables.*

Definition 3.5 (Multiple Substitution)**.** Let u be a proof term, $\boldsymbol{x} = x_0^{A_0}, \dots, x_n^{A_n}$ a sequence of variables and $v : A_0 \wedge \dots \wedge A_n$. The substitution $u^{v/\boldsymbol{x}} := u[v\,\pi_0/x_0^{A_0} \dots v\,\pi_n/x_n^{A_n}]$ replaces each variable $x_i^{A_i}$ of any term u with the ith projection of v.

The reduction rules for λ_{L}-terms are in Table 3. As usual, we adopt the reduction schema: $\mathcal{C}[t] \mapsto \mathcal{C}[u]$ whenever $t \mapsto u$ and for any context $\mathcal{C}$. With $\mapsto^*$ we shall denote the reflexive and transitive closure of the one-step reduction $\mapsto$.

Intuitionistic Reductions (application, projection and injection)

$$(\lambda x^A\, u)t \mapsto u[t/x^A] \qquad \langle u_0, u_1\rangle\, \pi_i \mapsto u_i \text{ for } i \in \{0,1\} \qquad \iota_i(t)[x_0.u_0, x_1.u_1] \mapsto u_i[t/x_i]$$

Case Distinction Permutations

$$t[x_0.u_0, x_1.u_1]\xi \mapsto t[x_0.u_0\xi, x_1.u_1\xi] \text{ if } \xi \text{ is a one-element stack}$$

Parallel Operator Permutations

$$_a(u_1 \parallel \dots \parallel u_m)\,\xi \mapsto {}_a(u_1\xi \parallel \dots \parallel u_m\xi) \text{ if } \xi \text{ is a stack of length 1 and } a \text{ does not occur free in } \xi$$

$$w\,_a(u_1 \parallel \dots \parallel u_m) \mapsto {}_a(wu_1 \parallel \dots \parallel wu_m), \text{ if } a \text{ does not occur free in } w$$

$$\lambda x^A\, {}_a(u_1 \parallel \dots \parallel u_m) \mapsto {}_a(\lambda x^A u_1 \parallel \dots \parallel \lambda x^A u_m)$$

$$\iota_i(\,{}_a(u_1 \parallel \dots \parallel u_m)) \mapsto {}_a(\iota_i(u_1) \parallel \dots \parallel \iota_i(u_m))$$

$$\langle {}_a(u_1 \parallel \dots \parallel u_m),\, w\rangle \mapsto {}_a(\langle u_1, w\rangle \parallel \dots \parallel \langle u_m, w\rangle), \text{ if } a \text{ does not occur free in } w$$

$$\langle w,\, {}_a(u_1\ \dots \parallel u_m)\rangle \mapsto {}_a(\langle w, u_1\rangle \parallel \dots \parallel \langle w, u_m\rangle), \text{ if } a \text{ does not occur free in } w$$

$$_a(u_1 \parallel \dots \parallel {}_b(w_1 \parallel \dots \parallel w_n) \dots \parallel u_m) \ \mapsto\ {}_b({}_a(u_1 \parallel \dots \parallel w_1 \parallel \dots \parallel u_m) \dots \parallel {}_a(u_1 \parallel \dots \parallel w_n \parallel \dots \parallel u_m))$$
$$\text{if } u_1, \dots, u_m \text{ and } {}_b(w_1 \parallel \dots \parallel w_n) \text{ do not contain active sessions}$$

Communication Reductions

Activation Reductions $\quad {}_a(u_1 \parallel \dots \parallel u_m) \ \mapsto\ {}_b(u_1[b/a] \parallel \dots \parallel u_m[b/a])$
where a is not active, b is a fresh *active* variable, and there is some occurrence of a in some u_i of the form aw, for a value w.

Basic Cross Reductions

$$\begin{array}{ll} {}_a(\mathcal{C}_1 \parallel \dots \parallel \mathcal{C}_i[a^{F_i \to G_i}\, t] \parallel \dots \parallel \mathcal{C}_j[a^{F_j \to G_j}\, u] \parallel \dots \parallel \mathcal{C}_m) & \text{for } F_i = G_j \\ \quad \mapsto\ {}_a(\mathcal{C}_1 \parallel \dots \parallel \mathcal{C}_i[a^{F_i \to G_i}\, t] \parallel \dots \parallel \mathcal{C}_j[t] \parallel \dots \parallel \mathcal{C}_m) & \\ {}_a(\mathcal{C}_1 \parallel \dots \parallel \mathcal{C}_j[a^{F_j \to G_j}\, u] \parallel \dots \parallel \mathcal{C}_i[a^{F_i \to G_i}\, t] \parallel \dots \parallel \mathcal{C}_m) & \text{for } F_j = G_i \\ \quad \mapsto\ {}_a(\mathcal{C}_1 \parallel \dots \parallel \mathcal{C}_j[t] \parallel \dots \parallel \mathcal{C}_i[a^{F_i \to G_i}\, t] \parallel \dots \parallel \mathcal{C}_m) & \end{array}$$

where a is active, the displayed occurrence of a is rightmost in the simply typed λ-terms $\mathcal{C}_i[a^{F_i \to G_i} t]$ and $\mathcal{C}_j[a^{F_j \to G_j} u]$, $1 \le i < j \le m$, $\mathcal{C}_i, \mathcal{C}_j$ are simple contexts and t does not contain free variables which are bound in $\mathcal{C}_i[a^{F_i \to G_i} t]$.

Simplification Reductions $\quad {}_a(u_1 \parallel \dots \parallel u_m) \mapsto u_{j_1} \parallel \dots \parallel u_{j_n}$ for $1 \le j_1 < \dots < j_n \le m$
if a does not occur in $u_{j_1}, \dots, u_{j_n}$

Cross Reductions $\quad {}_a(\mathcal{C}_1[a^{F_1 \to G_1}\, t_1] \parallel \dots \parallel \mathcal{C}_m[a^{F_m \to G_m}\, t_m]) \ \mapsto\ {}_b(s_1 \parallel \dots \parallel s_m)$
where a is active, $\mathcal{C}_j[a^{F_j \to G_j}\, t_j]$ for $1 \le j \le m$ are simply typed λ-terms; the displayed $a^{F_j \to G_j}$ is rightmost in each of them; b is a fresh inactive channel; for $1 \le i \le m$, we define

$$s_i = \begin{cases} {}_a(\mathcal{C}_1[a^{F_1 \to G_1}\, t_1] \parallel \dots \parallel \mathcal{C}_i[t_j^{\,b^{B_i \to B_j}\langle \boldsymbol{y}_i\rangle/\boldsymbol{y}_j}] \parallel \dots \parallel \mathcal{C}_m[a^{F_m \to G_m}\, t_m]) & \text{if } G_i \neq \bot \\ \\ {}_a(\mathcal{C}_1[a^{F_1 \to G_1}\, t_1] \parallel \dots \parallel \mathcal{C}_i[b^{B_i \to \bot}\langle \boldsymbol{y}_i\rangle] \parallel \dots \parallel \mathcal{C}_m[a^{F_m \to G_m}\, t_m]) & \text{if } G_i = \bot \end{cases}$$

where $F_j = G_i$ in the axiom schema if $G_i \neq \bot$; $\boldsymbol{y}_z$ for $1 \le z \le m$ is the sequence of the free variables of t_z bound in $\mathcal{C}_z[a^{F_z \to G_z}\, t_z]$; and B_z is the type of $\langle \boldsymbol{y}_z\rangle$.

Table 3: Reduction Rules for λ_{L} where $\mathcal{A} = (F_1 \to G_1) \vee \dots \vee (F_m \to G_m)\ \in\ \mathrm{L}$.

We explain the reduction rules of λ_{L} both from the computational and the proof-theoretic point of view.

Intuitionistic Reductions and Case Distinction Permutations.

These are the usual computational rules for the simply typed λ-calculus [21] representing the application of a function, the selection of an element from a pair and the choice of a case in a case distinction. The calculus also features permutations for the case distinction construct, which correspond to standard natural deduction $\vee$-permutations. From the logical point of view, all these are standard Prawitz reductions [34] for NJ.

An example of the correspondence between the evaluation of terms and proof-simplification is β-reduction in λ-calculus: $(\lambda x^A\, u)t \mapsto u[t/x]$, which corresponds to the following proof transformation:

$$\dfrac{\dfrac{\begin{matrix}[x^A : A]\\ \vdots \\ u : B\end{matrix}}{\lambda x^A\, u : A \to B} \qquad \begin{matrix}\vdots\\ t : A\end{matrix}}{(\lambda x^A\, u)t : B} \qquad \mapsto \qquad \begin{matrix}\vdots\\ t : A\\ \vdots\\ u[t/x] : B\end{matrix}$$

From a proof-theoretic perspective, this transformation avoids the use of the formula $A \to B$, which might violate the subformula property. The derivation on the left, to be reduced, is often called a *redex*. The inference steps introducing and eliminating $A \to B$ form a *detour*; they can indeed be considered an unnecessary deviation from a more direct way of deriving B.

Activation Reductions

These reductions enable us to transform a non-active channel which is applied to a value into an active channel. Only active channels can communicate.

Basic Cross and Simplification Reductions

The goal of these reductions is to implement communication, namely, to transmit programs in the form of simply typed λ-terms. Since λ_{L}-terms may contain more than one occurrence of a channel application, the first choice to make is which occurrence should contain the next output message. For example, here we have two communicating processes connected by the channel a:

$$(*) \qquad\qquad {}_{a}(\quad a(x(a\,s)) \quad \| \quad a(y(aw)) \quad)$$

where s, w are simply typed λ-terms not containing a. The first process $a(x(a\,s))$ contains two occurrences of the channel a. The channel application $a(x(a\,s))$ cannot transmit the message $x(a\,s)$, because the channel a might be used with a different type in the second process, so type preservation after reduction would fail. Therefore the only possibility here is to choose the second channel application $a\,s$ as the one containing the output message, in this case s. In general, to make sure that the message does not contain the channel a, it is enough to choose as application occurrence that contains the output message the *rightmost occurrence* of the channel a in the process.

The second choice is which occurrence of a channel application should receive the current message. For example, in the term $(*)$ above the second process $a(y(aw))$ contains two occurrences of a. Since a channel can both send and receive, and in particular, usually, first sends and then receives, we are led to choose again the rightmost occurrence of a channel application as the receiving one.

Continuing our example and summing up, we will have the following reduction:

$$(*) \qquad \mapsto \qquad {}_a(\ a(x(a\,s)) \ \parallel\ a(ys)\ )$$

As we can see, the message s in correspondence of the rightmost occurrence of a in $a(x(a\,s))$ is transmitted by the first process to the rightmost application of a in $a(y(aw))$.

The third choice to make is which processes should receive messages and which processes should send them. As anticipated in the previous section, we are guided by the typing. Let us consider for instance the term

$${}_a(\ x\,(a^{A\to B}\,s) \parallel\ y\,(a^{B\to A}\,t)\ )$$

where s and t are specific simply typed λ-terms, $x : B \to C$ and $y : A \to C$. A basic cross reduction rule corresponding to this typing rule admits communication in two directions, from left to right and from right to left, because looking at the types, the second process can receive messages from the first and vice versa. A reduction from left to right, for instance, is

$${}_a(\ x\,(a^{A\to B}\,s) \parallel\ y\,(a^{B\to A}\,t)\ ) \ \mapsto\ {}_a(\ x\,(a^{A\to B}\,s) \parallel\ y\,s\ )$$

whereas from right to left is

$${}_a(\ x\,(a^{A\to B}\,s) \parallel\ y\,(a^{B\to A}\,t)\ ) \ \mapsto\ {}_a(\ x\,t \parallel y\,(a^{B\to A}\,t)\ \ )$$

We can thus see that our reduction rules are naturally non-deterministic.

The fourth, and last, choice to make is what to do with the processes that do not contain any communication channel. The idea is that whenever a term contains no channel occurrence, it has already reached a result, as it does not need to interact with the context further. Hence at the end of the computation we may wish to select some of the processes that have reached their own results and consider them all together the global result of the computation. Thus we introduce simplification reductions. These reductions enable us to eliminate useless communication channels and to remove unnecessary duplicates generated by other reductions. Suppose, for example, that the process v in ${}_a(v \parallel w)$ does not contain occurrences of the channel a while the process w does. This intuitively means that w is waiting to communicate with v, but the communication is impossible. If this is the case, we simply apply the reduction ${}_a(v \parallel w) \mapsto v$ in order to remove the useless channel a and keep only the process v which does not need it.

We present some specific examples of basic cross reductions.

Example 3.1. Let us consider the axiom $\neg A \vee (\top \to A)$, which is an abbreviation of $(A \to \bot) \vee (\top \to A)$ and is equivalent to $\mathsf{EM} = A \vee \neg A$. The corresponding basic cross reduction is:

$${}_a(\mathcal{C}_0[a^{A\to\bot}t] \parallel \mathcal{C}_1[a^{\top\to A}u]) \mapsto {}_a(\mathcal{C}_0[at] \parallel \mathcal{C}_1[t])$$

Example 3.2. The basic cross reductions corresponding to the axiom $(A \to B) \vee (C \to A) \vee (B \to C)$ are:

$${}_a(\mathcal{C}_0[a^{A\to B}t] \parallel \mathcal{C}_1[a^{C\to A}u] \parallel u_2) \mapsto {}_a(\mathcal{C}_0[at] \parallel \mathcal{C}_1[t] \parallel u_2)$$

$${}_a(u_0 \parallel \mathcal{C}_1[a^{C\to A}t] \parallel \mathcal{C}_2[a^{B\to C}u]) \mapsto {}_a(u_0 \parallel \mathcal{C}_1[at] \parallel \mathcal{C}_2[t])$$

$${}_a(\mathcal{C}_0[a^{A\to B}u] \parallel u_1 \parallel \mathcal{C}_2[a^{B\to C}t]) \mapsto {}_a(\mathcal{C}_0[t] \parallel u_1 \parallel \mathcal{C}_2[at])$$

These reductions enable the process i to transmit the message t to the process $|i+1|_{(mod\ 3)}$.

From a proof-theoretic point of view, basic cross reductions correspond to a simple proof simplification. If we consider, for instance, the rule below on the left – corresponding to the axiom $\mathsf{EM} = \neg A \vee A$ – we have, similarly to [17], the basic cross reduction below on the right:

$$\dfrac{\begin{matrix}[\neg A]\\ \vdots\\ C\end{matrix} \qquad \begin{matrix}[A]\\ \vdots\\ C\end{matrix}}{C} \qquad\qquad \dfrac{\begin{matrix}\dfrac{[\neg A] \quad \overset{\delta}{A}}{\bot}\\ \vdots\\ C\end{matrix} \qquad \begin{matrix}[A]\\ \vdots\\ C\end{matrix}}{C} \quad\mapsto\quad \dfrac{\begin{matrix}\dfrac{[\neg A] \quad \overset{\delta}{A}}{\bot}\\ \vdots\\ C\end{matrix} \qquad \begin{matrix}\overset{\delta}{A}\\ \vdots\\ C\end{matrix}}{C}$$

if no assumption of δ is discharged below $\bot$ and above the bottommost C. From a computational perspective, δ corresponds to a message that does not depend on its computational environment and hence that can be transmitted by a basic cross reduction without violating type preservation. From the perspective of natural deduction, intuitively, the instance of the rule displayed in the reduction might be hiding some redex that should be reduced. The reduction precisely exposes this potential intuitionistic redex and we are thus able to reduce it. More instances of $\neg A$ and A might occur in the respective branches and have to be reduced. But if no formula is discharged by the rule for EM in some branch, we can eliminate that branch by a simplification reduction:

$$\dfrac{\overset{\gamma}{C} \qquad \overset{\theta}{C}}{C}\,1 \qquad\mapsto\qquad \overset{\theta}{C}$$

where no formula is discharged by the rule application 1 in θ.

Another simple example of basic cross reduction is given by the rule for the axiom $\mathsf{Lin} = (A \to B) \vee (B \to A)$:

$$\dfrac{\begin{matrix}[A\to B]\\ \vdots\\ C\end{matrix} \qquad \begin{matrix}[B\to A]\\ \vdots\\ C\end{matrix}}{C} \qquad \dfrac{\begin{matrix}\dfrac{[A\to B] \quad \overset{\delta}{A}}{B}\\ \vdots\\ C\end{matrix} \qquad \begin{matrix}\dfrac{[B\to A] \quad \overset{\theta}{B}}{A}\\ \vdots\\ C\end{matrix}}{C} \quad\mapsto\quad \dfrac{\begin{matrix}\dfrac{[A\to B] \quad \overset{\delta}{A}}{B}\\ \vdots\\ C\end{matrix} \qquad \begin{matrix}\overset{\delta}{A}\\ \vdots\\ C\end{matrix}}{C}$$

Here the conditions are still that no assumption of δ is discharged below B but above the bottommost C.

If we consider a generic axiom $\mathcal{A}$ we will have a rule application

$$\dfrac{\begin{matrix}[F_1\to G_1]\\ \vdots\\ B\end{matrix} \quad \ldots \quad \begin{matrix}\dfrac{[H\to G_i] \quad \overset{\delta}{H}}{G_i}\\ \vdots\\ B\end{matrix} \quad \ldots \quad \begin{matrix}\dfrac{[F_j\to H] \quad \overset{\theta}{F_j}}{H}\\ \vdots\\ B\end{matrix} \quad \ldots \quad \begin{matrix}[F_m\to G_m]\\ \vdots\\ B\end{matrix}}{B}\,(\mathcal{A})$$

where no assumption of δ is discharged below G_i but above the bottommost B, and by basic cross reduction we obtain

$$\dfrac{\begin{matrix}[F_1\to G_1]\\ \vdots\\ B\end{matrix} \quad \ldots \quad \begin{matrix}\dfrac{[H\to G_i] \quad \overset{\delta}{H}}{G_i}\\ \vdots\\ B\end{matrix} \quad \ldots \quad \begin{matrix}\overset{\delta}{H}\\ \vdots\\ B\end{matrix} \quad \ldots \quad \begin{matrix}[F_m\to G_m]\\ \vdots\\ B\end{matrix}}{B}\,(\mathcal{A})$$

Parallel Operator Permutations

These reductions regulate the interaction between parallel operators and the other computational constructs. The first two permutations in the table are the standard $\vee$-permutations needed for the subformula property already in NJ. The following four are used in the normalization proof.

The remaining permutation

$$
{}_a(u_1 \parallel \dots \parallel {}_b(w_1 \parallel \dots \parallel w_n) \dots \parallel u_m) \;\mapsto\; {}_b({}_a(u_1 \parallel \dots \parallel w_1 \parallel \dots \parallel u_m) \parallel \dots \parallel {}_a(u_1 \parallel \dots \parallel w_n \parallel \dots \parallel u_m))
$$

addresses the *scope extrusion* issue of private channels. For instance, if we consider the term ${}_b({}_a(v \parallel \mathcal{C}[b\,(\lambda x\, ax)]) \parallel w)$ we can see that $\mathcal{C}[b\,(\lambda x\, ax)]$ wishes to send the term $\lambda x\, ax$ containing the channel a to w along the channel b. This is not possible, though, since the channel a is private. This issue is solved in π-calculus using the congruence $\nu a(P \,|\, Q) \,|\, R \equiv \nu a(P \,|\, Q \,|\, R)$, provided that a does not occur in R, condition that can always be satisfied by α-conversion. Our logical system requires a different solution, which is not just permuting w inward but also duplicating it:

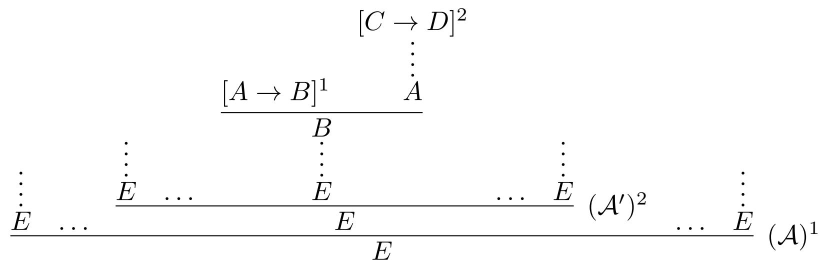
$$
{}_b({}_a(v \parallel \mathcal{C}[b\,(\lambda x\, ax)]) \parallel w) \mapsto {}_a({}_b(v \parallel w) \parallel {}_b(\mathcal{C}[b\,(\lambda x\, ax)] \parallel w))
$$

Now $\mathcal{C}[b\,(\lambda x\, ax)]$ can send $\lambda x\, ax$ to w. If b does not occur in v, we also simplify as follows:

$$
{}_a({}_b(v \parallel w) \parallel {}_b(\mathcal{C}[b\,(\lambda x\, ax)] \parallel w)) \mapsto {}_a(v \parallel {}_b(\mathcal{C}[b\,(\lambda x\, ax)] \parallel w))
$$

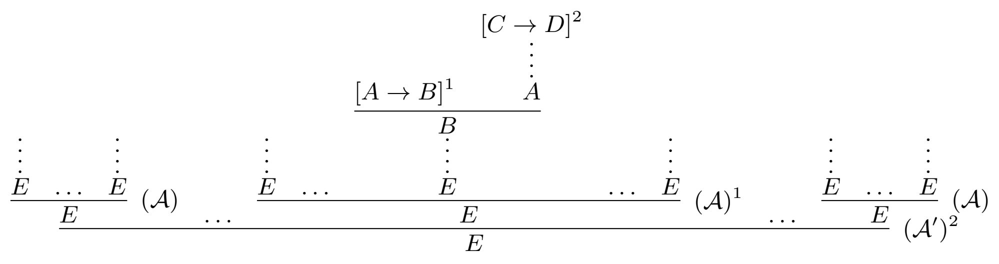
obtaining associativity of composition as in π-calculus. However, if b occurs in v, we cannot simplify and we keep the two copies of w to enable v and $\mathcal{C}[b\,(\lambda x\, ax)]$ to communicate with w.

From a proof-theoretical perspective, these reductions are needed when we have a derivation of the form



in which we need to simplify the rule application $(\mathcal{A})$ by a basic cross reduction or cross reduction. If we directly reduced $(\mathcal{A})$, we would move the derivation of A that depends on the hypothesis $C \to D$ above some other premiss of $(\mathcal{A})$. But after such a reduction, $C \to D$ cannot be discharged by $(\mathcal{A}')$ anymore. Hence, we permute $(\mathcal{A})$ upwards as follows:

$\mapsto$



If we reduce $(\mathcal{A})$ now, $(\mathcal{A}')$ can still discharge the hypothesis $C \to D$.

Cross Reductions

Basic cross reductions successfully implement the kind of communication that is used in actual parallel programming, but they are inadequate from the proof-theoretic point of view. As it turns out, they are not enough to guarantee that normal form proofs enjoy the subformula property. We introduce cross reductions to solve this problem. A proof-theoretical analysis of cross reductions is at page 16, but we first discuss the computational issues motivating them. The computational limit of basic cross reductions is that they can only transmit messages that do not have dependencies with their computational environment. Indeed, a message may contain free variables which are locally bound in the process, but in some case it is necessary to transmit the message anyway. Consider for example a particular active session ${}_{a}(\mathcal{C}[a\,u] \parallel \mathcal{D}[a\,t])$. If u did not depend on the computational environment $\mathcal{C}[\ ]$ we could use a basic cross reduction and send u as it is. In general, though, this is not possible. The problem is that the free variables $\boldsymbol{y}$ of u which are bound in $\mathcal{C}[a\,u]$ by some λ cannot be permitted to become free; otherwise, the connection between the binders $\lambda\,\boldsymbol{y}$ and the occurrences of the variables $\boldsymbol{y}$ would be lost and they could be no more replaced by actual values when the inputs for the $\lambda\,\boldsymbol{y}$ are available. For example, we could have $u = v\,\boldsymbol{y}$ and

$$\mathcal{C}[a\,u] = w\,(\lambda\boldsymbol{y}\,a\,(v\,\boldsymbol{y}))$$

and reducing as follows:

$${}_{a}(w\,(\lambda\boldsymbol{y}\,a\,(v\,\boldsymbol{y})) \parallel \mathcal{D}[a\,t]) \;\mapsto\; {}_{a}(w\,(\lambda\boldsymbol{y}\,a\,(v\,\boldsymbol{y})) \parallel \mathcal{D}[v\,\boldsymbol{y}])$$

would be computationally wrong since the term $v\,\boldsymbol{y}$ will have no access to the actual values of the variables $\boldsymbol{y}$ when they will become available to $\lambda\boldsymbol{y}\,a\,(v\,\boldsymbol{y})$.

These problems are typical of *process migration*, and can be solved by the concepts of *code mobility* [20] and *closure* [27]. Informally, code mobility is defined as the capability to dynamically change the bindings between code fragments and the locations where they are executed. Indeed, in order to be executed, a piece of code needs a computational environment and its resources, like data, program counters or global variables. In our case the context $\mathcal{C}[\ ]$ is the computational environment or *closure* of the process u and the variables $\boldsymbol{y}$ are the resources it needs. Now, moving a process outside its environment always requires extreme care: the bindings between a process and the environment resources must be preserved. This is the task of the *migration mechanisms*, which allow a migrating process to resume correctly its execution in the new location.

Our particular migration mechanism works by creating a duplicate of the original session for each process contained in it. One process for each copy of the session receives its message, but since the communication broke some of the computational dependencies of the message, a new communication channel b is established to connect the message to its old environment. The channel b enables the message to access the values of the free variables of the message which are bound in the original environment. Considering again our example and assuming that the free variables of t which are bound in $\mathcal{D}[a\,t]$ are $\boldsymbol{z}$ we have:

$${}_{a}(w\,(\lambda\boldsymbol{y}\,a\,(v\,\boldsymbol{y})) \parallel \mathcal{D}[a\,t]) \;\mapsto\; {}_{b}(\;{}_{a}(w\,(\lambda\boldsymbol{y}\,t^{b\,\langle\boldsymbol{y}\rangle/\boldsymbol{z}}) \parallel \mathcal{D}[a\,t]) \;\parallel\; {}_{a}(w\,(\lambda\boldsymbol{y}\,a\,(v\,\boldsymbol{y})) \parallel \mathcal{D}[v\,b\,\boldsymbol{z}])\;)$$

where b is typed in such a way that $b\,\boldsymbol{z}$ has the same type as $\boldsymbol{y}$ and $b\,\langle\boldsymbol{y}\rangle$ has the same type as $\boldsymbol{z}$.

The duplication of the original session produced by the cross reduction has a computational meaning which is strictly related to the non-determinism of basic cross reductions. Indeed, while a basic cross reduction is an individual communication, the full cross reduction produces the results of all possible individual communications, and keeps these results in parallel. Namely, for every possible pair of processes that can communicate, the reduction produces a copy of the original session in which the communication has been performed. Therefore each copy of the original session in the result of a cross reduction is exactly the result of one possible communication. Cross reduction thus makes the communication step deterministic. The simplification reductions that might follow a cross reduction will bring back non-determinism in the computation by discarding some of the produced results, and hence selecting one possible choice of individual communication.

Technically, the general reduction is:

$$_a(\mathcal{C}_1[a^{F_1 \to G_1}\, t_1] \parallel \ldots \parallel \mathcal{C}_m[a^{F_m \to G_m}\, t_m]) \;\mapsto\; _b(s_1 \parallel \ldots \parallel s_m)$$

where for $1 \leq i \leq m$

$$s_i = \begin{cases} _a(\mathcal{C}_1[a^{F_1 \to G_1}\, t_1] \ldots \parallel \mathcal{C}_i[t_j^{\,b^{B_i \to B_j}\langle \boldsymbol{y}_i\rangle / \boldsymbol{y}_j}] \parallel \ldots \parallel \mathcal{C}_m[a^{F_m \to G_m}\, t_m]) & \text{if } G_i \neq \bot \\ _a(\mathcal{C}_1[a^{G_1 \to G_1}\, t_1] \ldots \parallel \mathcal{C}_i[b^{B_i \to \bot}\langle \boldsymbol{y}_i\rangle] \parallel \ldots \parallel \mathcal{C}_m[a^{F_m \to G_m}\, t_m]) & \text{if } G_i = \bot \end{cases}$$

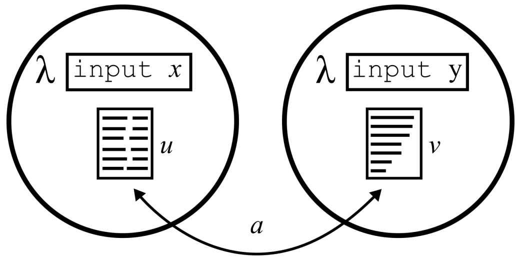
If we look at the reduction definition, the term $_b(s_1 \parallel \ldots \parallel s_m)$ encodes the results of all possible communications by the channel a. The unnecessary duplicates will be possibly reduced later on by simplification reductions. In particular, each process $\mathcal{C}_k[a^{F_k \to G_k}\, t_k]$ transmits its message t_k to the term $\mathcal{C}_l[a^{F_l \to G_l}\, t_l]$ such that $F_k = G_l$, which is the designated receiver of the message. If for some n, it holds $G_k = F_n$, the sender $\mathcal{C}_k[a^{F_k \to G_k}\, t_k]$ will also receive a message t_n from another process, and will reduce to $\mathcal{C}_k[t_n^{b_k\langle \boldsymbol{y}_k\rangle / \boldsymbol{y}_n}]$. Here the free variables $\boldsymbol{y}_n$ of the message t_n are replaced by the application of the fresh communication channel b_k, which has the task to receive the values of $\boldsymbol{y}_n$ when they will be forwarded. If, on the other hand, $G_k = \bot$, the sender $\mathcal{C}_k[a^{F_k \to G_k}\, t_k]$ is not a receiver and it will reduce to $\mathcal{C}_k[b_k\langle \boldsymbol{y}_k\rangle]$ since there is no incoming message. In either case, the applications $b_k\langle \boldsymbol{y}_k\rangle$ will be used to forward the values of the variables $\boldsymbol{y}_k$ if and when they will be available. Technically, the redex reduces to the term $_b(s_1 \parallel \ldots \parallel s_m)$ where s_p for $p \in \{1, \ldots, m\}$ is a copy of the original redex and contains only one process of the form $\mathcal{C}_k[t_n^{b_k\langle \boldsymbol{y}_k\rangle / \boldsymbol{y}_n}]$ or $\mathcal{C}_k[b_k\langle \boldsymbol{y}_k\rangle]$ resulting from the communication. We also require that the channel occurrences used in the communication are rightmost in the processes containing them. This guarantees the preservation of the bindings of channel variables.

Example 3.3. The specific cross reduction rule corresponding to the axiom $\mathsf{Lin} = (A \to B) \vee (B \to A)$ is the following:

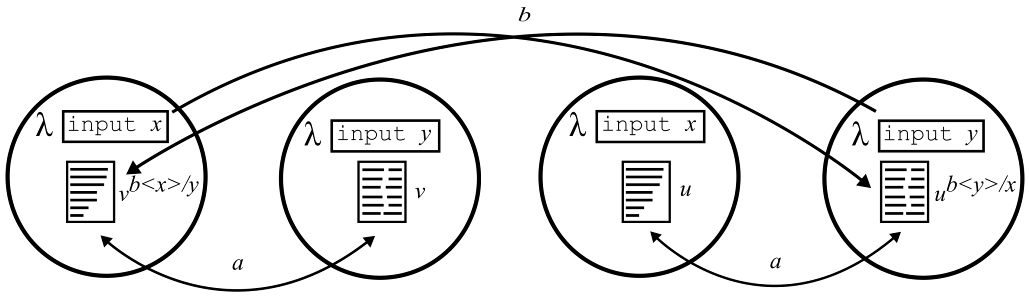
$$\mathcal{C}[a^{A \to B} u] \parallel_a \mathcal{D}[a^{B \to A} v] \quad \mapsto \quad (\mathcal{C}[v^{b\langle \boldsymbol{x}\rangle / \boldsymbol{y}}] \parallel_a \mathcal{D}[av]) \;\parallel_b\; (\mathcal{C}[au] \parallel_a \mathcal{D}[u^{b\langle \boldsymbol{y}\rangle / \boldsymbol{x}}])$$

where $\boldsymbol{x}$ is the tuple of free variables of u which are bound in $\mathcal{C}[au]$ and $\boldsymbol{y}$ is the tuple of free variables of v which are bound in $\mathcal{D}[av]$.

We can schematically represent the term $\mathcal{C}[a^{A \to B} u] \parallel_a \mathcal{D}[a^{B \to A} v]$ as



After the reduction we obtain the following term:



In the latter term, b reconnects the messages u and v to their original environments by the substitutions $b\langle \boldsymbol{y}\rangle/\boldsymbol{x}$ and $b\langle \boldsymbol{x}\rangle/\boldsymbol{y}$ respectively. Thus, when available, $\boldsymbol{x}$ will be sent to u and $\boldsymbol{y}$ to v through the channel b.

A concrete example of the application of this cross reduction rule on a simple term is

$$_a(\lambda x\, a(zx) \parallel \lambda y\, \langle y, a(yk)\rangle) \;\mapsto\; {}_b({}_a(\lambda x\, bxk \parallel \lambda y\, \langle y, a(yk)\rangle) \parallel {}_a(\lambda x\, a(zx) \parallel \lambda y\, \langle y, (z(by))\rangle))$$

where z, k are variables, $a : A \to B$ in $\lambda x\, a(zx)$, $a : B \to A$ in $\lambda y\, \langle y, a(yk)\rangle$. We use the color blue for the new terms produced by the cross reduction.

For a proof-theoretic view of cross reductions, we first illustrate the reduction for the rule (Lin) where Lin $= (A \to B) \vee (B \to A)$. Consider the derivation

$$\dfrac{\begin{array}{c}\dfrac{[A \to B] \quad \begin{array}{c}\Gamma\\ \gamma\\ A\end{array}}{B}\\ \vdots\\ C\end{array} \qquad \begin{array}{c}\dfrac{[B \to A] \quad \begin{array}{c}\Delta\\ \delta\\ B\end{array}}{A}\\ \vdots\\ C\end{array}}{C}$$

in which Γ and Δ are the hypotheses of γ and δ that are respectively discharged below B and A. Assume that $A \to B$ does not occur in γ and $B \to A$ does not occur in δ as hypotheses discharged by the rule for Lin. A cross reduction transforms this derivation into

$$\dfrac{\dfrac{\begin{array}{c}\dfrac{[A \to B] \quad \begin{array}{c}\Gamma\\ \gamma\\ A\end{array}}{B}\,{}_1\\ \vdots\\ C\end{array} \qquad \begin{array}{c}\begin{array}{c}\Delta\\ \hline\hline \Gamma\end{array}\,{}_3\\ \gamma\\ A\\ \vdots\\ C\end{array}}{C}\,{}_1 \qquad \dfrac{\begin{array}{c}\begin{array}{c}\Gamma\\ \hline\hline \Delta\end{array}\,{}_3\\ \delta\\ B\\ \vdots\\ C\end{array} \qquad \begin{array}{c}\dfrac{[B \to A] \quad \begin{array}{c}\Delta\\ \delta\\ B\end{array}}{A}\,{}_2\\ \vdots\\ C\end{array}}{C}\,{}_2}{C}\,{}_3$$

where the notation $\begin{array}{c}X\\ \hline\hline Y\end{array}$ means that we derive each occurrence of element D of Y as follows:

$$\begin{array}{c} [\bigwedge X \to \bigwedge Y]^3 \quad \begin{array}{c} X\\ \hline\hline \bigwedge X\\ \hline \bigwedge Y\end{array}\,\wedge i\\ \hline\hline D\end{array}\,\wedge e$$

where $\bigwedge X$ and $\bigwedge Y$ are the conjunctions of the elements of X and Y respectively, $\dfrac{X}{\bigwedge X}\wedge i$ is the obvious derivation by conjunction introductions, and $\dfrac{\bigwedge Y}{D}\wedge e$ is the obvious derivation of D by conjunction eliminations. Notice that this reduction duplicates the original rule application once for each of its branches, and removes only one discharged formula from each copy of the original application: this is exactly what happens in the general case.

Finally, suppose we have a generic application of $(\mathcal{A})$, below left, in which all Γ_i for $1 \le i \le m$ are discharged between G_i and B. It reduces by cross reduction as follows:

$$\dfrac{\begin{matrix}\dfrac{[F_1 \to G_1]^* \qquad \begin{matrix}\Gamma_1\\ \alpha_1\\ F_1\end{matrix}}{G_1}\\ \vdots \\ B\end{matrix} \qquad \dots \qquad \begin{matrix}\dfrac{[F_m \to G_m]^* \qquad \begin{matrix}\Gamma_m\\ \alpha_m\\ F_m\end{matrix}}{G_m}\\ \vdots \\ B\end{matrix}}{B}\ * \qquad \mapsto \qquad \dfrac{\delta_1 \quad \dots \quad \delta_m}{B}\ **$$

$$\text{with } \delta_i = \quad \dfrac{\begin{matrix}\dots\\ B\end{matrix} \quad \dots \quad \begin{matrix}\dfrac{[\bigwedge\Gamma_i \to \bigwedge\Gamma_j]^{**} \qquad \dfrac{\Gamma_i}{\bigwedge\Gamma_i}\wedge i}{\dfrac{\bigwedge\Gamma_j}{\alpha_j}\wedge e}\\ G_i\\ \vdots\\ B\end{matrix} \quad \dots \quad \begin{matrix}\dots\\ B\end{matrix}}{B}\ *$$

Here $i \in \{1, \dots, m\}$, α_j is the derivation of the premise $F_j = G_i$ associated by $\mathcal{A}$ to G_i, and a double inference line denotes a derivation only containing applications of the named rule.

Cross reductions are complex but can be used in a relatively simple way in practice, as shown by the following example.

Example 3.4. Consider the term

$${}_a(\ a\,(\lambda x\, xt) \parallel {}_b(\lambda z\, z(b(uz)\pi_0) \parallel a * (\lambda y\, b\langle s, y\rangle))) \tag{2}$$

where the channels are typed as follows: $a : \neg A$ in the leftmost process and $a : \top \to A$ in the rightmost, $b : B \to C$ in the second process and $b : C \to B$ in the rightmost, and $* : \top$. The channel b cannot transmit $\langle s, y\rangle$ without first solving the issue of y losing connection with its binder. However, after communicating the message $(\lambda x\, xt)$ by a basic cross reduction through the channel a, (2) would become ${}_a(\ a\,(\lambda x\, xt) \parallel {}_b(\lambda z\, z(b(uz)\pi_0) \parallel (\lambda x\, xt)(\lambda y\, b\langle s, y\rangle))\ )$ and then ${}_a(\ a\,(\lambda x\, xt) \parallel {}_b(\lambda z\, z(b(uz)\pi_0) \parallel b\langle s, t\rangle)\ )$ and now the third process can send its message $\langle s, t\rangle$ to the second without any issue, obtaining

$${}_a(\ a\,(\lambda x\, xt) \parallel {}_b(\lambda z\, z(\langle s, t\rangle\pi_0) \parallel b\langle s, t\rangle)\ ) \mapsto^* \lambda z\, zs$$

However, this is not the only possible communication order: instead of using basic cross reductions, we can use fully general cross reductions and directly use the channel b to send the term $\langle s, y\rangle$ to the second process without waiting for the channel a:

$$\begin{array}{c}
{}_a(\ a(\lambda x\, xt) \parallel {}_b(\lambda z\, z(b(uz)\pi_0) \parallel a * (\lambda y\, b\langle s, y\rangle))\ )\\
\mapsto\ {}_a(\ a(\lambda x\, x\, t) \parallel {}_c(\ {}_b(\lambda z\, z(\langle s, cz\rangle\pi_0) \parallel a * (\lambda y\, b\langle s, y\rangle)) \parallel {}_b(\lambda z\, z(b(uz)\pi_0) \parallel a * (\lambda y\, u(cy))))\ )\\
\mapsto\ {}_a(\ a(\lambda x\, x\, t) \parallel {}_c(\ {}_b(\lambda z\, zs \parallel a * (\lambda y\, b\langle s, y\rangle)) \parallel {}_b(\lambda z\, z(b(uz)\pi_0) \parallel a * (\lambda y\, u(cy))))\ )\\
\mapsto\ {}_a(\ a(\lambda x\, x\, t) \parallel {}_c(\ \lambda z\, zs \parallel {}_b(\lambda z\, z(b(uz)\pi_0) \parallel a * (\lambda y\, u(cy))))\ )\\
\mapsto^*\ {}_a(\ a(\lambda x\, x\, t) \parallel {}_c(\lambda z\, zs \parallel a * (\lambda y\, u(cy)))\ )\ \mapsto\ {}_a(\ a(\lambda x\, x\, t) \parallel \lambda z\, zs)\ \mapsto \lambda z\, zs
\end{array}$$

We only apply a cross reduction to a term $_a(u_1 \parallel \ldots \parallel u_m)$ when the channel a is *active*. Ultimately, a channel can be activated only if one of its occurrences is directly applied to a value or may receive a value to be transmitted sometimes in the future. Intuitively, an active channel has or will have an argument that needs to be transmitted because it might produce new computations in the receiving process. This activation condition would be natural in a call-by-value reduction strategy, but here we are not transmitting messages according to a call-by-value policy just for the sake of it. An activation condition is indeed *necessary*, because unrestricted cross reductions do not always terminate. For example, reducing

$$_a(\lambda y^B\, a^{B\to B}\, y \parallel x^{(C\to C)\to B\to B}\, (\lambda z^C\, a^{C\to C} z))$$

as follows generates a loop:

$$\mapsto\ _b(_a((\lambda y\, b\, y \parallel x\,(\lambda z\, a z))) \parallel {}_a((\lambda y\, a\, y \parallel x\, \lambda z\, b z)))\ \mapsto^*\ _b(\lambda y\, b\, y \parallel x\, \lambda z\, b z)\ \mapsto \ldots$$

We present now some examples of computations. The first one shows that some instances of λ_{L} are strictly more expressive than simply typed λ-calculus.

Example 3.5 (Parallel OR). As is well known there is no λ-term $\mathsf{O} : \mathsf{Bool} \to \mathsf{Bool} \to \mathsf{Bool}$ such that we have $\mathsf{OFF} \mapsto \mathsf{F}$, $\mathsf{O}u\mathsf{T} \mapsto \mathsf{T}$, $\mathsf{OT}u \mapsto \mathsf{T}$ without evaluating u. O can instead be defined in Boudol's calculus [10], λ_{G} [2] and λ_{CL} [3] We add the boolean type in our calculus and we define λ_{EM} as the instance of λ_{L} based on the axiom $\mathsf{EM} = \neg A \vee A$. Then the term for parallel OR is

$$\mathsf{O} := \lambda x^{\mathsf{Bool}}\, \lambda y^{\mathsf{Bool}}(\ \ (\mathsf{if}\, x\, \mathsf{then}\, \mathsf{T}\, \mathsf{else}\, (a\mathsf{F})\, \mathsf{efq}_{\mathsf{Bool}}) \quad \parallel_a \quad (\mathsf{if}\, y\, \mathsf{then}\, \mathsf{T}\, \mathsf{else}\, a\, \mathsf{T})\ \ )$$

where $a : \mathsf{Bool} \to \bot$, $a : \top \to \mathsf{Bool}$, $\mathsf{T} : \top$ and "$\mathsf{if}\, u\, \mathsf{then}\, s\, \mathsf{else}\, t$" is as usual. Now

$$\mathsf{O}\, u\, \mathsf{T} \mapsto^* (\mathsf{if}\, u\, \mathsf{then}\, \mathsf{T}\, \mathsf{else}\, (a\mathsf{F})\, \mathsf{efq}_{\mathsf{Bool}}) \parallel_a (\mathsf{if}\, \mathsf{T}\, \mathsf{then}\, \mathsf{T}\, \mathsf{else}\, a\, \mathsf{T}) \mapsto^* (\mathsf{if}\, u\, \mathsf{then}\, \mathsf{T}\, \mathsf{else}\, (a\mathsf{F})\, \mathsf{efq}_{\mathsf{Bool}}) \parallel_a \mathsf{T} \ \mapsto\ \mathsf{T}$$

And symmetrically $\mathsf{O}\,\mathsf{T}\, u \ \mapsto^*\ \mathsf{T}$. On the other hand

$$\mathsf{O}\,\mathsf{F}\,\mathsf{F} \mapsto^* (\mathsf{if}\, \mathsf{F}\, \mathsf{then}\, \mathsf{T}\, \mathsf{else}\, (a\mathsf{F})\, \mathsf{efq}_{\mathsf{Bool}}) \parallel_a (\mathsf{if}\, \mathsf{F}\, \mathsf{then}\, \mathsf{T}\, \mathsf{else}\, a\, \mathsf{T}) \mapsto^* (a\mathsf{F})\, \mathsf{efq}_{\mathsf{Bool}} \parallel_a a \ \mapsto\ \mathsf{F}$$

Notice that even if all λ_{L}-terms reduce to a normal form in a finite number of steps, the existence of a term behaving as a parallel OR is not obvious. Indeed, one of the arguments of the term might be a variable and never reduce neither to T nor to F. For instance, during the reduction of $\mathsf{O}u\mathsf{T}$ above, if $u = x$ then the $\mathsf{if\ then\ else}$ construct occurring in $(\mathsf{if}\, u\, \mathsf{then}\, \mathsf{T}\, \mathsf{else}\, (a\mathsf{F})\, \mathsf{efq}_{\mathsf{Bool}}) \parallel_a \mathsf{T}$ could never reduce.

We now present an example which simulates the interactions of an online sale.

Example 3.6 (Buyer and Vendor). We model the following transaction: a buyer tells a vendor a product name $\mathsf{prod} : \mathsf{String}$, the vendor computes the monetary cost $\mathsf{price} : \mathbb{N}$ of prod and communicates it to the buyer, the buyer sends back the credit card number $\mathsf{card} : \mathsf{String}$ which is used to pay.

We introduce the following functions: $\mathsf{cost} : \mathsf{String} \to \mathbb{N}$ with input a product name prod and output its cost price; $\mathsf{pay\_for} : \mathbb{N} \to \mathsf{String}$ with input a price and output a credit card number card; $\mathsf{use} : \mathsf{String} \to \mathbb{N}$ that produces money using as input a credit card number $\mathsf{card} : \mathsf{String}$. The buyer and the vendor are the contexts $\mathcal{B}$ and $\mathcal{V}$ of type Bool. The communication channel a is typed using the instance $(\mathsf{String} \to \mathbb{N}) \vee (\mathbb{N} \to \mathsf{String})$ of the axiom $\mathsf{Lin} = (A \to B) \vee (B \to A)$.

The program is:

$$\begin{aligned}
&\mathcal{B}[a(\mathsf{pay\_for}(a(\mathsf{prod})))] \parallel_a \mathcal{V}[\mathsf{use}(a(\mathsf{cost}(a\, 0)))] \\
&\mapsto \mathcal{B}[a(\mathsf{pay\_for}(a(\mathsf{prod})))] \parallel_a \mathcal{V}[\mathsf{use}(a(\mathsf{cost}(\mathsf{prod})))] \\
&\mapsto \mathcal{B}[a(\mathsf{pay\_for}(a(\mathsf{prod})))] \parallel_a \mathcal{V}[\mathsf{use}(a(\mathsf{price}))] \\
&\mapsto \mathcal{B}[a(\mathsf{pay\_for}(\mathsf{price}))] \parallel_a \mathcal{V}[\mathsf{use}(a(\mathsf{price}))] \\
&\mapsto\ \mathcal{B}[a(\mathsf{card})] \parallel_a \mathcal{V}[\mathsf{use}(a(\mathsf{price}))] \ \mapsto\ \mathcal{B}[a(\mathsf{card})] \parallel_a \mathcal{V}[\mathsf{use}(\mathsf{card})]
\end{aligned}$$

Finally $\mapsto \mathcal{V}[\mathsf{use}(\mathsf{card})]$: the buyer has performed its duty and the vendor uses the card number to obtain the due payment.

Remark 3.2 (**Ambient Calculus and Cross Reductions**). The ambient calculus [13] is a formalism where processes are represented as ambients in which the computation is carried out. Ambients' structure is similar to the structure of processes in process calculi, but instead of communicating, ambients can enter, exit, or dissolve the boundaries of other ambients. Even though the issues addressed by ambient calculus seem close to those that motivate the introduction of cross reductions in λ_{L}, the operations of ambients are assumed not to break any computational dependence, while cross reductions are meant to handle changes of environment which violate computational dependencies by restoring such dependencies.

The proof below that reductions preserve types is standard.

Theorem 3.1 (Subject Reduction). *If $t : A$ and $t \mapsto u$, then $u : A$ and all the free variables of u appear among those of t.*

Proof. Since the argument for intuitionistic reductions is standard, that for simplification reductions and permutations is trivial and that for basic cross reductions is just a simpler version of that for cross reductions, we only consider the latter case. Suppose then that

$$_a(\mathcal{C}_1[a^{F_1 \to G_1}\, t_1] \parallel \ldots \parallel \mathcal{C}_m[a^{F_m \to G_m}\, t_m]) \;\mapsto\; _b(s_1 \parallel \ldots \parallel s_m)$$

Suppose that s_i for $1 \le i \le m$ is

$$_a(\mathcal{C}_1[a^{F_1 \to G_1}\, t_1] \parallel \ldots \parallel \mathcal{C}_i[t_j^{b^{B_i \to B_j}\langle \boldsymbol{y}_i\rangle / \boldsymbol{y}_j}] \parallel \ldots \parallel \mathcal{C}_m[a^{F_m \to G_m}\, t_m])$$

where $G_i \neq \bot$. Assuming that $\langle \boldsymbol{y}_i\rangle : B_i$, the terms $b^{B_i \to B_j}\langle \boldsymbol{y}_i\rangle$ are correct. Hence the term $t_j^{b^{B_i \to B_j}\langle \boldsymbol{y}_i\rangle / \boldsymbol{y}_j}$, by Definition 3.5, is correct as well. Now, some of the assumptions of the reduction rule are that $t_j^{b^{B_i \to B_j}\langle \boldsymbol{y}_i\rangle / \boldsymbol{y}_j}$ has the same type as $a^{F_i \to G_i}\, t_i$; and that $\boldsymbol{y}_i$ for $1 \le i \le m$ is the sequence of the free variables of t_i which are bound in $\mathcal{C}_i[a^{F_i \to G_i}\, t_i]$. Hence, by construction, all the variables $\boldsymbol{y}_i$ are bound in each $\mathcal{C}_i[t_j^{b^{B_i \to B_j}\langle \boldsymbol{y}_i\rangle / \boldsymbol{y}_j}]$. Since, moreover, a is rightmost in each $\mathcal{C}_i[a^{F_i \to G_i}\, t_i]$ and b is fresh, no new free variable is created.

Suppose that, on the other hand, s_i for $1 \le i \le m$ is

$$_a(\mathcal{C}_1[a^{F_1 \to G_1}\, t_1] \parallel \ldots \parallel \mathcal{C}_i[b^{B_i \to \bot}\langle \boldsymbol{y}_i\rangle] \parallel \ldots \parallel \mathcal{C}_m[a^{F_m \to G_m}\, t_m])$$

where $G_i = \bot$. Assuming that $\langle \boldsymbol{y}_i\rangle : B_i$, the term $b^{B_i \to \bot}\langle \boldsymbol{y}_i\rangle$ is a correct term. Now, since $G_i = \bot$, the term $b^{B_i \to \bot}\langle \boldsymbol{y}_i\rangle$ has the same type as $a^{F_i \to G_i}\, t_i$. Since $\boldsymbol{y}_i$ for $1 \le i \le m$ is the sequence of the free variables of t_i which are bound in $\mathcal{C}_i[a^{F_i \to G_i}\, t_i]$ and since b is fresh, no new free variable is created. $\square$

Similarly to what happens in the embedding of hypersequent proofs into 2-systems [15], the starting point of our normalization strategy will be to rewrite any λ_{L}-term into a parallel composition of simply-typed λ-terms, formally defined below.

Definition 3.6 (Parallel Form). A *parallel form* is defined inductively as follows: a simply typed λ-term is a parallel form; if $u_1, \ldots, u_m$ are parallel forms, then both $_a(u_1 \parallel \ldots \parallel u_m)$ and $u_1 \parallel \ldots \parallel u_m$ are parallel forms.

Definition 3.7 (Normal Forms and Normalizable Terms).

- A *redex* is a term u such that $u \mapsto v$ for some v and basic reduction of Table 3. A term t is called a *normal form* or, simply, *normal*, if there is no t' such that $t \mapsto t'$. We define NF to be the set of normal λ_{L}-terms.

- A sequence, finite or infinite, of proof terms $u_1, u_2, \ldots, u_n, \ldots$ is said to be a reduction of t, if $t = u_1$, and for all i, $u_i \mapsto u_{i+1}$. A proof term u of λ_{L} is *normalizable* if there is a finite reduction of u whose last term is a normal form.

4 The Normalization Theorem

In this section, we shall prove the normalization theorem for λ_{L}: every proof term of λ_{L} reduces in a finite number of steps to a normal form. By subject reduction, this implies that the corresponding natural deduction proofs normalize. We remark that the permutations between communications are necessary to solve the scope extrusion problem, but at the same time, they undermine strong normalization, because they enable silly loops, like those generated by cyclically permuting cuts over cuts in cut-elimination for sequent calculi.

According to our normalization strategy, we start from any term and reduce it in parallel form by Proposition 4.2. Then we cyclically interleave three reduction phases: an *intuitionistic phase* in which we reduce all intuitionistic redexes, an *activation phase* in which we activate all sessions that can be activated, and a *communication phase* in which the active sessions exchange all messages (cross reductions) and the processes extract information from the received messages (projections and case distinction permutations).

Proving that this strategy terminates is not easy, as two kinds of loops might occur:
1. Intuitionistic reductions can generate new activable channels that need to transmit messages, while message exchanges can generate new intuitionistic reductions.
2. During the communication phase new sessions may be generated after each cross reduction and old sessions may be duplicated after each permutation. Each of these sessions may need to send new messages, for instance to forward some message just received, and hence the count of active sessions might increase forever and the communication phase never terminate.

We avoid the first loop by exploiting the complexity of the exchanged messages. Since messages are *values*, we shall define a notion of *value complexity* (Definition 4.5), which will simultaneously ensure that: (i) after reducing an intuitionistic redex which is not a projection or a case distinction permutation, the new active sessions only transmit messages with value complexity smaller than the complexity of such redex; (ii) after transmitting a message, all newly generated intuitionistic redexes do not have complexity greater than the value complexity of such message. Proposition 4.6 will settle the matter, but in turn requires a series of preparatory lemmas. Namely, we shall study in Lemma 4.5 and Lemma 4.6 how arbitrary substitutions affect the value complexity of a term; then we shall determine in Lemma 4.8 and Lemma 4.7 how case reductions affect the value complexity.

We prove that the second loop is not possible by showing in Lemma 4.10 that the transmission of messages, during the communication phase, cannot produce new active sessions. Intuitively, the newly generated channels and the old duplicated ones are *frozen*, in the sense that only the reduction of an intuitionistic redex can activate them and in doing it will generate communication redexes with smaller complexity than the redex itself.

We define now a recursive normalization algorithm that represents the constructive content of the normalization proof. The master reduction strategy has three phases: in the first we reduce all possible intuitionistic redexes; in the second we activate all possible sessions; in the third we use the reduction relation $\succ$ defined below. By $\succ$ we men that we permute an uppermost active session $_{a}(u_1 \parallel \dots \parallel u_m)$ until all u_i for $1 \leq i \leq m$ are simply typed λ-terms and then apply the cross reductions followed by projections and case permutations.

Definition 4.1 (Side Reduction Strategy)**.** Let t be a term and $_{a}(u_1 \parallel \dots \parallel u_m)$ be an active session occurring in t such that no active session occurs in u or v. We write $\quad t \succ t' \quad$ whenever t' has been obtained from t by applying one of the following to $_{a}(u_1 \parallel \dots \parallel u_m)$:

1. a permutation reduction

$$_{a}(u_1 \parallel \dots \parallel {}_{b}(w_1 \parallel \dots \parallel w_n) \dots \parallel u_m) \;\mapsto\; {}_{b}({}_{a}(u_1 \parallel \dots \parallel w_1 \dots \parallel u_m) \parallel \dots \parallel {}_{a}(u_1 \parallel \dots \parallel w_n \dots \parallel u_m))$$

   if $u_i = {}_{b}(w_1 \parallel \dots \parallel w_n)$ for some $1 \leq i \leq m$;

2. a cross reduction, if $u_1, \ldots, u_m$ are intuitionistic terms, immediately followed by projections and case distinction permutations inside the newly generated simply typed λ-terms;

3. a simplification reduction $_a(u_1 \parallel \ldots \parallel u_m) \mapsto u_{j_1} \parallel \ldots \parallel u_{j_n}$, for $1 \leq j_1 < \ldots < j_n \leq m$, if a does not occur in $u_{j_1}, \ldots, u_{j_n}$

Definition 4.2 (Master Reduction Strategy)**.** Let t be any term not in normal form. We transform it into a term u in parallel form and execute the following three-step procedure.

1. *Intuitionistic Phase.* As long as u contains intuitionistic redexes, we reduce them.

2. *Activation Phase.* As long as u contains activation redexes, we reduce them.

3. *Communication Phase.* As long as u contains active sessions, we apply the Side Reduction Strategy (Definition 4.1) to u, then we go to step 1.

We start by defining the value complexity of messages. It represents a measure of the complexity of the redexes that a message can generate after transmission. For terms of the form λxu and $\iota_i(u)$, which are values in the usual sense, such complexity is defined on the type of the term. For pairs $\langle u, v\rangle$ and case distinctions $t[x.u, y.v]$, which we consider containers for what is usually taken to be a value, we recursively pick the maximum among the value complexities of u and v. This is a crucial point. If we used types as value complexities for pairs and case distinctions, our normalization argument would break down when new channels are generated during cross reductions: their type can be more complex than the type of the original channel and any hope of finding a decreasing measure would be shattered.

Definition 4.3 (Complexity of a Type)**.** The complexity of a type T is defined as follows:

- if $T = \bot$ or $T = P$ for a propositional variable P, then the complexity of T is 0;
- if $T = A_1 \to A_2$, $T = A_1 \wedge A_2$ or $T = A_1 \vee A_2$ and the complexity of A_i is n_i, then the complexity of T is $n_1 + n_2 + 1$.

Definition 4.4 (Case-free)**.** A stack $\sigma = \sigma_1\sigma_2 \ldots \sigma_n$ is *case-free* if σ_i is not of the form $[z_1.w_1, z_2.w_2]$ for any $i \in \{1, \ldots, n\}$.

Definition 4.5 (Value Complexity)**.** For any simply typed λ-term $s : T$, the value complexity of s is defined as the first case that applies among the following:

- if $s = \lambda xu$, $s = \iota_i(u)$, then the value complexity of s is the complexity of its type T;
- if $s = \langle u, v\rangle$, then the value complexity of s is the maximum among those of u and v.
- if $s = t[x.u, y.v]\sigma$ where σ is case-free, then the value complexity of s is the maximum among the value complexities of $u\sigma$ and $v\sigma$;
- otherwise, the value complexity of s is 0.

Recall that values are supposed to be those terms that can either generate an intuitionistic redex when plugged into another term or become a term with that capability, like an active channel acting as the endpoint of a transmission. The value complexity of a term, as expected, never exceeds the complexity of its type.

Proposition 4.1. *Let $u : T$ be any simply typed λ-term. Then the value complexity of u is not greater than the complexity of T.*

Proof. By induction on u. There are several cases, according to the shape of u.

- If u is of the form $\lambda x w$, $\iota_i(w)$, then the value complexity of u is indeed the complexity of T.
- If u is of the form $\langle v_1, v_2\rangle$ then, by the induction hypothesis, the value complexities of v_1 and v_2 are at most the complexity of their respective types T_1 and T_2, and hence at most the complexity of $T = T_1 \wedge T_2$, and we are done.
- If u is of the form $v_0[z_1.v_1, z_2.v_2]$ then, by induction hypothesis, the value complexities of v_1 and v_2 are at most the complexity of T, and we are done.
- In all other cases, the value complexity of u is 0, which trivially satisfies the thesis.

□

The complexity of an intuitionistic redex $t\xi$, where ξ is a stack of length 1, is defined as the value complexity of t.

Definition 4.6 (Complexity of the Intuitionistic Redexes)**.** Let r be an intuitionistic redex. The complexity of r is defined as follows:

- If $r = (\lambda x u)t$, then the complexity of r is the type of $\lambda x u$.
- If $r = \iota_i(t)[x.u, y.v]$, then the complexity of r is the type of $\iota_i(t)$.
- if $r = \langle u, v\rangle\pi_i$ then the complexity of r is the value complexity of $\langle u, v\rangle$.
- if $r = t[x.u, y.v]\xi$, where ξ is a stack of length 1, then the complexity of r is the value complexity of $t[x.u, y.v]$.

The value complexity is used to define the complexity of communication redexes as well. Intuitively, it is the value complexity of the most complex message ready to be transmitted.

Definition 4.7 (Complexity of the Communication Redexes)**.**

- The *complexity of a channel occurrence* $a\langle t_1, \ldots, t_n\rangle$ of a channel a is the value complexity of $\langle t_1, \ldots, t_n\rangle$.
- The *complexity of a communication redex* $_a(u_1 \parallel \ldots \parallel u_m)$ is the maximum among the complexities of the occurrences of a in $u_1, \ldots, u_m$.
- The *complexity of a permutation redex* $_a(u_1 \parallel \ldots \parallel {}_b(w_1 \parallel \ldots \parallel w_n) \ldots \parallel u_m)$ is 0.

As our normalization strategy suggests, application and injection redexes should be treated differently from the others, because they generate the real computations.

Definition 4.8. We distinguish two groups of redexes:

- **Group 1**: Application and injection redexes.
- **Group 2**: Communication, projection and case distinction permutation redexes.

Before proving the required auxiliary results, we summarize the steps of the normalization procedure used in the proof of Proposition 4.12. The procedure for a term t is the following:

1. We reduce t to a term t_1 in parallel form (Proposition 4.2).

2. We start the induction on the maximal complexity τ of the redexes in t_1.
3. We reduce all intuitionistic redexes in t_1 and obtain t_2 in which the complexity of redexes is still smaller than or equal to τ (Proposition 4.9 which requires Lemmas 4.4 and 4.8).
4. We reduce all activation redexes and obtain t_3 in which the complexity of redexes is still smaller than or equal to τ (Lemma 4.3).
5. We fire all cross reductions and obtain t_4 which only contains redexes of Group 1 with complexity smaller than or equal to τ (Lemma 4.11 which requires Lemma 4.10).
6. If t_4 contains redexes of Group 1, we reduce them. The redexes of Group 2 that we thus create have complexity smaller than τ (Proposition 4.9.2).
7. After recursively eliminating all redexes of Group 1, we obtain a term t_5 which has only redexes of complexity smaller than τ, thus we can use the induction hypothesis.

We show that any term can be reduced to a parallel form.

Proposition 4.2 (Parallel Form)**.** *Let $t : A$ be any term. Then $t \mapsto^* t'$, with t' parallel form.*

Proof. By induction on t. As a shortcut, if a term u reduces to a term u' that can be denoted as u'' omitting parentheses, we write $u \rightrightarrows^* u''$. We present here only the interesting cases. The unabridged version of the proof is in the appendix (A.1).

- $t = u\,v$. By the induction hypothesis,

$$u \rightrightarrows^* u_1 \parallel u_2 \parallel \dots \parallel u_{n+1}$$

$$v \rightrightarrows^* v_1 \parallel v_2 \parallel \dots \parallel v_{m+1}$$

and each term u_i and v_i, for $1 \le i \le n+1, m+1$, is a simply typed λ-term. Applying several permutations we obtain

$$\begin{aligned} t &\rightrightarrows^* (u_1 \parallel u_2 \parallel \dots \parallel u_{n+1})\,v \\ &\rightrightarrows^* u_1\,v \parallel u_2\,v \parallel \dots \parallel u_{n+1}\,v \\ &\rightrightarrows^* u_1\,v_1 \parallel u_1\,v_2 \parallel \dots \parallel u_1\,v_{m+1} \parallel \dots \\ &\qquad \dots \parallel u_{n+1}\,v_1 \parallel u_{n+1}\,v_2 \parallel \dots \parallel u_{n+1}\,v_{m+1} \end{aligned}$$

- $t = s[x.u, y.v]$. By the induction hypothesis,

$$\begin{aligned} s &\rightrightarrows^* s_1 \parallel s_2 \parallel \dots \parallel s_{n+1} \\ u &\rightrightarrows^* u_1 \parallel u_2 \parallel \dots \parallel u_{m+1} \\ v &\rightrightarrows^* v_1 \parallel v_2 \parallel \dots \parallel v_{p+1} \end{aligned}$$

and each term s_i for $1 \le i \le n+1$, u_j for $1 \le j \le m+1$, and v_k for $1 \le k \le p+1$ is a simply typed λ-term. Applying several permutations we obtain

$$\begin{aligned}
t \rightrightarrows^* & s_1[x.u, y.v] \parallel s_2[x.u, y.v] \parallel \ldots \parallel s_{n+1}[x.u, y.v] \\
\rightrightarrows^* & s_1[x.u_1, y.v] \parallel s_1[x.u_2, y.v] \parallel \ldots \parallel s_1[x.u_{m+1}, y.v] \parallel \\
& s_2[x.u_1, y.v] \parallel s_2[x.u_2, y.v] \parallel \ldots \parallel s_2[x.u_{m+1}, y.v] \parallel \\
& \ldots \parallel s_{n+1}[x.u_1, y.v] \parallel s_{n+1}[x.u_2, y.v] \parallel \ldots \parallel s_{n+1}[x.u_{m+1}, y.v] \\
\rightrightarrows^* & s_1[x.u_1, y.v_1] \parallel s_1[x.u_1, y.v_2] \parallel \ldots \parallel s_1[x.u_1, y.v_{p+1}] \parallel \\
& s_1[x.u_2, y.v_1] \parallel s_1[x.u_2, y.v_2] \parallel \ldots \parallel s_1[x.u_2, y.v_{p+1}] \parallel \\
& \ldots \parallel s_1[x.u_{m+1}, y.v_1] \parallel s_1[x.u_{m+1}, y.v_2] \parallel \ldots \parallel s_1[x.u_{m+1}, y.v_{p+1}] \\
& s_2[x.u_1, y.v_1] \parallel s_2[x.u_1, y.v_2] \parallel \ldots \parallel s_2[x.u_1, y.v_{p+1}] \parallel \\
& s_2[x.u_2, y.v_1] \parallel s_2[x.u_2, y.v_2] \parallel \ldots \parallel s_2[x.u_2, y.v_{p+1}] \parallel \\
& \ldots \parallel s_2[x.u_{m+1}, y.v_1] \parallel s_2[x.u_{m+1}, y.v_2] \parallel \ldots \parallel s_2[x.u_{m+1}, y.v_{p+1}] \\
& s_{n+1}[x.u_1, y.v_1] \parallel s_{n+1}[x.u_1, y.v_2] \parallel \ldots \parallel s_{n+1}[x.u_1, y.v_{p+1}] \parallel \\
& s_{n+1}[x.u_2, y.v_1] \parallel s_{n+1}[x.u_2, y.v_2] \parallel \ldots \parallel s_{n+1}[x.u_2, y.v_{p+1}] \parallel \\
& \ldots \parallel s_{n+1}[x.u_{m+1}, y.v_1] \parallel s_{n+1}[x.u_{m+1}, y.v_2] \parallel \ldots \parallel s_{n+1}[x.u_{m+1}, y.v_{p+1}].
\end{aligned}$$

□

The following lemma shows that the activation phase of our reduction strategy is finite.

Lemma 4.3 (Activate!). *Let t be any term in parallel form that does not contain intuitionistic redexes and whose communication redexes have complexity at most τ. Then there exists a finite sequence of activation reductions that results in a term t' that contains no redexes, except cross reduction redexes of complexity at most τ.*

Proof. The proof is by induction on the number n of subterms of the form $_a(u_1 \parallel \ldots \parallel u_m)$ of t which are not active sessions. If there are no activation redexes in t, the statement trivially holds. Assume there is at least one activation redex $r = {_a}(v_1 \parallel \ldots \parallel v_m)$. We apply an activation reduction to r and obtain a term r' with $n-1$ subterms of the form $_a(u_1 \parallel \ldots \parallel u_m)$ which are not active sessions. In order to apply the induction hypothesis on r', which yields the claim, we only need to verify that all communication redexes of r' have complexity at most τ.

For, let c be any channel variable which is bound in r'. Since r' is obtained from r by renaming the non-active bound channel variable a to an active one b, every occurrence of c in r' is of the form $(c\,t)[b/a]$ for some subterm $c\,t$ of r. Thus $c\,t[b/a] = c[b/a]\langle t_1[b/a], \ldots, t_n[b/a]\rangle$, where each t_i is not a pair. It suffices to show that the value complexity of $t_i[b/a]$ is exactly the value complexity of t_i. We proceed by induction on the size of t_i. We can write $t_i = r_1\,\sigma$, where σ is a case-free stack. If r_1 is of the form $\lambda x w$, $\langle q_1, q_2\rangle$, $\iota_i(w)$, x, dw, with d channel variable, then the value complexity of $t_i[b/a]$ is the same as that of t_i; note that if $r_1 = \langle q_1, q_2\rangle$, then σ is not empty. If $r_1 = v_0[x_1.v_1, x_2.v_2]$, then σ is empty, otherwise s would contain a permutation redex. Hence, the value complexity of $t_i[b/a]$ is the maximum among the value complexities of $v_1[b/a]$ and $v_2[b/a]$. By the induction hypothesis, their value complexities are those of v_1 and v_2, hence the value complexity of $t_i[b/a]$ is the same as that of t_i, which concludes the proof. □

We show a simple property of the value complexity that we will need later.

Lemma 4.4 (Why Not 0?). *Let u be any simply typed λ-term and σ be a non-empty case-free stack. Then the value complexity of $u\sigma$ is 0.*

Proof. By induction on the size of u.

- If u is of the form $(\lambda x.w)\rho$, $\iota_i(w)\rho$, $\langle v_1, v_2\rangle\rho$, $a\rho$ or $x\rho$, where ρ is case-free, then $u\sigma$ has value complexity 0.
- If u is of the form $w\,\mathsf{efq}_P$, then P is atomic, thus σ must be empty, contrary to the assumptions.
- If u is of the form $v_0[z_1.v_1, z_2.v_2]\rho$, with ρ case-free, then by the induction hypothesis the value complexities of $v_1\rho\sigma$ and $v_2\rho\sigma$ are 0 and since the value complexity of $u\sigma$ is the maximum among them, $u\sigma$ has value complexity 0.

□

In order to formally study the effects of reducing a redex, we consider simple substitutions that just replace some occurrences of a term with another, allowing capture of variables. In practice, it will always be clear from the context which occurrences are replaced. These substitutions model the kind of transformations that are performed when reducing a redex: the redex is replaced by a new term, but with capture of variables.

Definition 4.9 (Simple Replacement)**.** By $s\{t/u\}$ we denote a term obtained by replacing *some* occurrences of a subterm u of s with a term t of the same type as u, possibly with capture of variables.

An alternative definition we could have adopted uses the familiar context notation: if $s = \mathcal{C}[u][u]\dots[u]$, then $s\{t/u\} = \mathcal{C}[t][t]\dots[t]$, which clearly shows that not all occurrences of u are replaced by t. Since our notation is simpler, we adopt it in place of this one based on contexts.

We now show an essential fact: the value complexity of $w\{v/s\}$ either remains at most as it was before the substitution or becomes exactly the value complexity of v.

Lemma 4.5 (The Change of Value)**.** *Let w, s, v be simply typed λ-terms with value complexity respectively θ, τ, τ'. Then the value complexity of $w\{v/s\}$ is either at most θ or equal to τ'. Moreover, if $\tau' \le \tau$, then the value complexity of $w\{v/s\}$ is at most θ.*

Proof. By induction on the size of w.

- $w\{v/s\} = x\,\{v/s\}$. We have two cases.
  - $s = x$. Then the value complexity of $w\{v/s\} = v$ is τ'. Moreover, if $\tau' \le \tau$, since $w = x = s$, we have $\theta = \tau$, thus $\tau' \le \theta$.
  - $s \ne x$. The value complexity of $w\{v/s\}$ is θ and we are done.
- $w\{v/s\} = \lambda xu\,\{v/s\}$. We have two cases.
  - $\lambda xu\,\{v/s\} = \lambda x(u\{v/s\})$. Then the value complexity of $w\{v/s\}$ is θ and we are done.
  - $\lambda xu\,\{v/s\} = v$. Then the value complexity of $w\{v/s\}$ is τ'. Moreover, if $\tau' \le \tau$, since $w = \lambda xu = s$, we have $\theta = \tau$, thus $\tau' \le \theta$.
- $w\{v/s\} = \langle q_1, q_2\rangle\,\{v/s\}$. We have two cases.
  - $\langle q_1, q_2\rangle\,\{v/s\} = \langle q_1\{v/s\}, q_2\{v/s\}\rangle$. Then by the induction hypothesis the value complexities of $q_1\{v/s\}$ and $q_2\{v/s\}$ are at most θ or equal to τ'. Since the value complexity of $w\{v/s\}$ is the maximum among the value complexities of $q_1\{v/s\}$ and $q_2\{v/s\}$, we are done.
  - $\langle q_1, q_2\rangle\,\{v/s\} = v$. Then the value complexity of $w\{v/s\}$ is τ'. Moreover, if $\tau' \le \tau$, since $w = \langle q_1, q_2\rangle = s$, we have $\theta = \tau$, thus $\tau' \le \theta$.

- $w\{v/s\} = \iota_i(u)\,\{v/s\}$. We have two cases.
  - $\iota_i(u)\,\{v/s\} = \iota_i(u\{v/s\})$. Then the value complexity of $w\{v/s\}$ is θ and we are done.
  - $\iota_i(u)\,\{v/s\} = v$. Then the value complexity of $w\{v/s\}$ is τ'. Moreover, if $\tau' \leq \tau$, since $w = \iota_i(u) = s$, we have $\theta = \tau$, thus $\tau' \leq \theta$.

- - $w\{v/s\} = (v_0[z_1.v_1, z_2.v_2])\rho\{v/s\} = v_0\{v/s\}[z_1.v_1\{v/s\}, z_2.v_2\{v/s\}](\rho\{v/s\})$, where ρ is a case-free stack. If ρ is not empty, then by Lemma 4.4, $v_1\{v/s\}\rho\{v/s\}$ and $v_2\{v/s\}\rho\{v/s\}$ have value complexity 0, so $w\{v/s\}$ has value complexity $0 \leq \theta$ and we are done. So assume ρ is empty. By the induction hypothesis applied to $v_1\{v/s\}$ and $v_2\{v/s\}$, the value complexity of $w\{v/s\}$ is at most θ or equal to τ' and we are done. Moreover, if $\tau' \leq \tau$, then by the induction hypothesis, the value complexity of $v_i\{v/s\}$ for $i \in \{1, 2\}$ is at most θ. Hence, the value complexity of $(v_0[z_1.v_1, z_2.v_2])\rho\{v/s\}$ is at most θ and we are done.
  - $w\{v/s\} = (v_0[z_1.v_1, z_2.v_2]\rho)\{v/s\} = v\rho_j\{v/s\} \ldots \rho_n\{v/s\}$, where $\rho = \rho_1 \ldots \rho_n$ is a case-free stack and $1 \leq j \leq n$. If $\rho_j \ldots \rho_n$ is not empty, then by Lemma 4.4, the value complexity of $w\{v/s\}$ is $0 \leq \theta$, and we are done. So assume $\rho_j \ldots \rho_n$ is empty. Then the value complexity of $w\{v/s\}$ is τ'. Moreover, if $\tau' \leq \tau$, since it must be $s = v_0[z_1.v_1, z_2.v_2]$, we have that the value complexity of $w\{v/s\} = v$ is $\tau' \leq \tau = \theta$.

- In all other cases, $w\{v/s\} = (r\,\rho)\{v/s\}$ where ρ is a case-free non-empty stack and r is of the form $\lambda x u$, $\langle q_1, q_2\rangle$, $\iota_i(u)$, x, or au. We distinguish three cases.
  - $(r\,\rho)\{v/s\} = r\{v/s\}\rho\{v/s\}$ and $r \neq s$. Then by Lemma 4.4 the value complexity of $w\{v/s\}$ is $0 \leq \theta$ and we are done.
  - $(r\,\rho)\{v/s\} = v\rho_j\{v/s\} \ldots \rho_n\{v/s\}$ given that $r = s$, $\rho = \rho_1 \ldots \rho_n$, and $1 \leq j \leq n$. Then by Lemma 4.4 the value complexity of $w\{v/s\}$ is $0 \leq \theta$ and we are done.
  - $r\rho\{v/s\} = v$. Then the value complexity of $w\{v/s\}$ is τ'. Moreover, if $\tau' \leq \tau$, since $w = r\rho = s$, we have $\theta = \tau$, thus $\tau' \leq \theta$.

□

Lemma 4.6 studies how the complexity of redexes in a term changes with simple replacements.

Lemma 4.6 (Replace!)**.** *Let u be a term in parallel form, v, s be any simply typed λ-terms, τ be the value complexity of v and τ' be the maximum among the complexities of the channel occurrences in v. Then every redex in $u\{v/s\}$ is either (i) already in v, (ii) of the form $r\{v/s\}$ and has complexity smaller than or equal to the complexity of some redex r of u, or (iii) has complexity τ or is a communication redex of complexity at most τ'.*

Proof. We prove the following stronger statement.

$(*)$ Every redex and channel occurrence in $u\{v/s\}$ it is either (i) already in v, (ii) of the form $r\{v/s\}$ or $aw\{v/s\}$ and has complexity smaller than or equal to the complexity of some redex r or channel occurrence aw of u, or (iii) has complexity τ or τ' or is a communication redex of complexity at most τ'.

We reason by induction on the size and by cases on the possible shapes of the term u. Only some interesting cases are shown here. See A.2, in the appendix, for the unabridged version of the proof.

- $w_0[z_1.w_1, z_2.w_2]\sigma\{v/s\}$, where σ is any case-free stack. By the induction hypothesis, $(*)$ holds for $w_0\{v/s\}$, $w_1\{v/s\}$, $w_2\{v/s\}$ and $\sigma_i\{v/s\}$ for $1 \leq i \leq n$. If

  $$w_0[z_1.w_1, z_2.w_2]\sigma\{v/s\} = \\ w_0\{v/s\}[z_1.w_1\{v/s\}, z_2.w_2\{v/s\}](\rho\{v/s\})$$

  we first observe that by Lemma 4.5, the value complexity of $w_0\{v/s\}$ is at most that of w_0 or exactly τ, hence the possible injection or case distinction permutation redex

  $$w_0\{v/s\}[z_1.w_1\{v/s\}, z_2.w_2\{v/s\}]$$

  satisfies the thesis. Again by Lemma 4.5, the value complexities of $w_1\{v/s\}$ and $w_2\{v/s\}$ are respectively at most that of w_1 and w_2 or exactly τ. Hence the complexity of the possible case distinction permutation redex

  $$(w_0[z_1.w_1\{v/s\}, z_2.w_2\{v/s\}])\sigma_1\{v/s\}$$

  is either τ, and we are done, or at most the value complexity of one among w_1, w_2, thus at most the value complexity of the case distinction permutation redex $(w_0[z_1.w_1, z_2.w_2])\sigma_1$ and we are done.

  If $w_0[z_1.w_1, z_2.w_2]\sigma\{v/s\} = v\,(\sigma_i\{v/s\})\dots(\sigma_n\{v/s\})$, then there could be a new intuitionistic redex, when $v = \lambda y\, q$, $v = \langle q_1, q_2\rangle$, $v = \iota_i(q)$ or $v = q_0[y_1.q_1, y_2.q_2]$. But the complexity of such a redex is τ.

- $a\,t\,\sigma\{v/s\}$, where a is a channel variable, t a term and $\sigma = \sigma_1 \dots \sigma_n$ is any case-free stack. By the induction hypothesis, $(*)$ holds for $t\{v/s\}, \sigma_i\{v/s\}$ where $1 \leq i \leq n$.

  If $a\,t\,\sigma\{v/s\} = v\,(\sigma_i\{v/s\})\dots(\sigma_n\{v/s\})$, then $v\,(\sigma_i\{v/s\})$ could be an intuitionistic redex, when $v = \lambda y\, w$, $v = \langle w_1, w_2\rangle$, $v = \iota_i(w)$ or $v = w_0[y_1.w_1, y_2.w_2]$. But the complexity of such a redex is equal to τ.

  If $a\,t\,\sigma\{v/s\} = a\,(t\{v/s\})(\sigma\{v/s\})$, in order to verify the thesis it is enough to check the complexity of the channel occurrence $a\,(t\{v/s\})$. By Lemma 4.5, the value complexity of $t\{v/s\}$ is at most the value complexity of t or exactly τ, thus either (ii) or (iii) holds.

- $_a(t_1 \parallel \dots \parallel t_m)\{v/s\}$. By the induction hypothesis, $(*)$ holds for $t_i\{v/s\}$ where $1 \leq i \leq m$. The only redex in $_a(t_1 \parallel \dots \parallel t_m)\{v/s\}$ and not in some $t_i\{v/s\}$ can be $_a(t_1\{v/s\} \parallel \dots \parallel t_m\{v/s\})$ itself. But the complexity of such redex equals the maximal complexity of the channel occurrences of the form aw occurring in some $t_i\{v/s\}$, hence it is τ, at most τ' or equal to the complexity of $_a(t_1 \parallel \dots \parallel t_m)$.

$\square$

We study now the complexity of the redexes generated by the reduction of injections.

Lemma 4.7 (Eliminate the Case!). *Let u be a term in parallel form. Then for any redex r in $u\{w_i[t/x_i]/\iota_i(t)[x_1.w_1, x_2.w_2]\}$ of complexity θ, either $\iota_i(t)[x_1.w_1, x_2.w_2]$ has complexity greater than θ; or there is a redex in u of complexity θ which belongs to the same group as r or is a case distinction permutation redex.*

Proof. Let $v = w_i[t/x_i]$ and $s = \iota_i(t)[x_1.w_1, x_2.w_2]$. We prove a stronger statement:

$(*)$ For any redex r in $u\{v/s\}$ of complexity θ, either $\iota_i(t)[x_1.w_1, x_2.w_2]$ has complexity greater than θ; or there is a redex in u of complexity θ which belongs to the same group as r or is a case distinction permutation redex. Moreover, for any channel occurrence in $u\{v/s\}$ with complexity

θ', either $\iota_i(t)[x_1.w_1, x_2.w_2]$ has complexity greater than θ', or there is an occurrence of the same channel with complexity greater than or equal to θ'.

The proof is by induction on the size of u and by cases according to its shape. We present here only the interesting cases. The unabridged version of the proof is in the appendix (A.3).

- $v_0[z_1.v_1, z_2.v_2]\sigma\{v/s\}$, where σ is any case-free stack. By the induction hypothesis, $(*)$ holds for $v_0\{v/s\}$, $v_1\{v/s\}$, $v_2\{v/s\}$ and $\sigma_i\{v/s\}$ for $1 \le i \le n$. If

  $$v_0[z_1.v_1, z_2.v_2]\sigma\{v/s\} = \\ v_0\{v/s\}[z_1.v_1\{v/s\}, z_2.v_2\{v/s\}](\rho\{v/s\})$$

  by Lemma 4.5 the value complexity of $w_i[t/x_i]$ is either at most the value complexity of w_i or the value complexity of t. In the first case, the value complexity of $w_i[t/x_i]$ is at most the value complexity of w_i which is at most the value complexity of $\iota_i(t)[x_1.w_1, x_2.w_2]$. Thus, by Lemma 4.5 the value complexities of $v_0\{v/s\}$, $v_1\{v/s\}$, $v_2\{v/s\}$ are at most the value complexity respectively of v_0, v_1, v_2. Hence, the complexity of the possible case distinction permutation redex

  $$(v_0[z_1.v_1\{v/s\}, z_2.v_2\{v/s\}])\sigma_1\{v/s\}$$

  is at most the complexity of $v_0[z_1.v_1, z_2.v_2]\sigma_1$ and we are done. Moreover, the possible injection or case distinction permutation redex

  $$v_0\{v/s\}[z_1.v_1\{v/s\}, z_2.v_2\{v/s\}]$$

  satisfies the thesis. In the second case, the value complexities of $v_0\{v/s\}$, $v_1\{v/s\}$, $v_2\{v/s\}$ are at most the value complexities of v_0, v_1, v_2 respectively or exactly the value complexity of t. Hence the complexity of the possible case distinction permutation redex $(v_0[z_1.v_1\{v/s\}, z_2.v_2\{v/s\}])\sigma_1\{v/s\}$ is either at most the value complexity of v_1, v_2, and we are done, or exactly the value complexity of t, which by Proposition 4.1 is at most the complexity of the type of t, thus is smaller than the complexity of the injection redex $\iota_i(t)[x_1.w_1, x_2.w_2]$ occurring in u. Moreover, the possible injection or case distinction permutation redex

  $$v_0\{v/s\}[z_1.v_1\{v/s\}, z_2.v_2\{v/s\}]$$

  has complexity equal to the value complexity of v_0 or the value complexity of t, and we are done again.

  If $v_0[z_1.v_1, z_2.v_2]\sigma\{v/s\} = v\,(\sigma_i\{v/s\})\dots(\sigma_n\{v/s\})$, then there could be a new intuitionistic redex, when $v = \lambda y\, q$, $v = \langle q_1, q_2\rangle$, $v = \iota_i(q)$ or $v = q_0[y_1.q_1, y_2.q_2]$. If the value complexity of $v = w_i[t/x_i]$ is at most the value complexity of w_i, then the complexity of $w_i[t/x_i](\sigma_i\{v/s\})$ is equal to the complexity of the permutation redex $(\iota_i(t)[x_1.w_1, x_2.w_2])\sigma_i$. If the value complexity of $v = w_i[t/x_i]$ is the value complexity of t, by Proposition 4.1 the complexity of $w_i[t/x_i](\sigma_i\{v/s\})$ is at most the complexity of the type of t, thus is smaller than the complexity of the injection redex $\iota_i(t)[x_1.w_1, x_2.w_2]$ occurring in u and we are done.

- $a\, t'\, \sigma\{v/s\}$, where a is a channel variable, t' a term and $\sigma = \sigma_1 \dots \sigma_n$ is any case-free stack. By induction hypothesis, $(*)$ holds for t' and $\sigma_i\{v/s\}$ for $1 \le i \le n$. Since $s \neq a\, t'\, \sigma_1 \dots \sigma_j$, the case $a\, t'\, \sigma\{v/s\} = v\,(\sigma_i\{v/s\})\dots(\sigma_n\{v/s\})$ is impossible.

  If $a\, t'\, \sigma\{v/s\} = a\,(t'\{v/s\})(\sigma\{v/s\})$, in order to verify the thesis it is enough to check the complexity of the channel occurrence $a\,(t'\{v/s\})$. By Lemma 4.5, the value complexity of $w_i[t/x_i]$ is either at most the value complexity of w_i or exactly the value complexity of t. In the first case, the value complexity of $w_i[t/x_i]$ is at most the value complexity of w_i which is at most the value complexity of $\iota_i(t)[x_1.w_1, x_2.w_2]$. Thus, by Lemma 4.5, the value complexity of $t'\{v/s\}$ is at most the value complexity of t' and we are done. In the second case, the value complexity of $t'\{v/s\}$ is the value complexity of t, which by Proposition 4.1 is at most the complexity of the type of t, thus smaller than the complexity of the injection redex $\iota_i(t)[x_1.w_1, x_2.w_2]$ occurring in u, which is what we wanted to show.

- ${}_a(t_1 \parallel \dots \parallel t_m)\{v/s\}$. By the induction hypothesis, $(*)$ holds for $t_i\{v/s\}$ for $1 \leq i \leq m$. The only redex in ${}_a(t_1 \parallel \dots \parallel t_m)\{v/s\}$ and not in some $t_i\{v/s\}$ can be ${}_a(t_1\{v/s\} \parallel \dots \parallel t_m\{v/s\})\{v/s\}$ itself. But the complexity of such redex equals the maximal complexity of the occurrences of the channel a in the $t_i\{v/s\}$. Hence the statement follows.

$\square$

We analyze now the complexity of the redexes generated by case distinction permutations.

Lemma 4.8 (In the Case)**.** *Let u be a term in parallel form. Then for any redex r_1 of Group 1 in $u\{t[x_1.v_1\xi, x_2.v_2\xi]/t[x_1.v_1, x_2.v_2]\xi\}$, where ξ is a stack of length 1, there is a redex in u with complexity greater than or equal to r_1; for any redex r_2 of Group 2 in $u\{t[x_1.v_1\xi, x_2.v_2\xi]/t[x_1.v_1, x_2.v_2]\xi\}$, there is a redex of Group 2 in u with complexity greater than or equal to r_2.*

Proof. Let $v = t[x_1.v_1\xi, x_2.v_2\xi]$ and $s = t[x_1.v_1, x_2.v_2]\xi$. We prove the following stronger statement.

$(*)$ For any redex r_1 of Group 1 in $u\{t[x_1.v_1\xi, x_2.v_2\xi]/t[x_1.v_1, x_2.v_2]\xi\}$ there is a redex in u with complexity greater than or equal to r_1; for any redex r_2 of Group 2 in $u\{t[x_1.v_1\xi, x_2.v_2\xi]/t[x_1.v_1, x_2.v_2]\xi\}$ there is a redex of Group 2 in u with complexity greater than or equal to r_2. Moreover, for any channel occurrence in $u\{v/s\}$ with complexity θ', there is in u an occurrence of the same channel with complexity greater than or equal to θ'.

We first observe that the possible Group 1 redexes $v_1\xi$ and $v_2\xi$ have at most the complexity of the case distinction permutation $t[x_1.v_1, x_2.v_2]\xi$. The rest of the proof is by induction on the size of u.

Only some interesting cases are shown here. See A.4, in the appendix, for the unabridged version of the proof.

- $w_0[z_1.w_1, z_2.w_2]\sigma\{v/s\}$, where σ is any case-free stack. By the induction hypothesis, $(*)$ holds for $w_0\{v/s\}$, $v_1\{v/s\}$, $v_2\{v/s\}$ and $\sigma_i\{v/s\}$ for $1 \leq i \leq n$. If

$$\begin{aligned} &w_0[z_1.w_1, z_2.w_2]\sigma\{v/s\} = \\ &w_0\{v/s\}[z_1.w_1\{v/s\}, z_2.w_2\{v/s\}](\sigma\{v/s\}) \end{aligned}$$

Since the value complexity of $t[x_1.v_1\xi, x_2.v_2\xi]$ is equal to the value complexity of $t[x_1.v_1, x_2.v_2]\xi$, by Lemma 4.5 the value complexities of $w_0\{v/s\}$, $w_1\{v/s\}$ and $w_2\{v/s\}$ are respectively at most that of w_0, w_1 and w_2. The complexity of the possible case distinction permutation redex $(w_0\{v/s\}[z_1.w_1\{v/s\}, z_2.w_2\{v/s\}])\sigma_1\{v/s\}$ is thus at most the value complexity of w_1, w_2 respectively, and hence the complexity of the case permutation redex $(w_0[z_1.w_1, z_2.w_2])\sigma_1$. Moreover, the possible injection or case permutation redex

$$w_0\{v/s\}[z_1.w_1\{v/s\}, z_2.w_2\{v/s\}]$$

has complexity equal to the value complexity of w_0 and we are done.

If $w_0[z_1.w_1, z_2.w_2]\sigma\{v/s\} = v\,(\sigma_i\{v/s\})\ldots(\sigma_n\{v/s\})$, then there could be a new case distinction permutation redex because $v = t[x_1.v_1\xi, x_2.v_2\xi]$. If ξ is case free, by Lemma 4.4, this redex has complexity 0 and we are done; if not, it has the same complexity as $t[x_1.v_1, x_2.v_2]\xi\sigma_1$ and we are done again.

- $a\,t\,\sigma\{v/s\}$, where a is a channel variable, t a term and $\sigma = \sigma_1\ldots\sigma_n$ is any case-free stack. By induction hypothesis, $(*)$ holds for t and $\sigma_i\{v/s\}$ where $1 \le i \le n$.

  If $a\,t\,\sigma\{v/s\} = a\,(t\{v/s\})(\sigma\{v/s\})$, in order to verify the thesis it is enough to check the complexity of the channel occurrence $a\,(t\{v/s\})$. Since the value complexity of $t[x_1.v_1\xi, x_2.v_2\xi]$ is equal to the value complexity of $t[x_1.v_1, x_2.v_2]\xi$, by Lemma 4.5 the value complexity of $t\{v/s\}$ is at most that of t, and we are done.

  If $a\,t\,\sigma\{v/s\} = a\,t\,[x_1.v_1\xi, x_2.v_2\xi]\,(\sigma_i\{v/s\})\ldots(\sigma_n\{v/s\})$ then $v\,(\sigma_i\{v/s\})$ can be a new permutation redex. If ξ is case free, this redex has complexity 0 and we are done. Otherwise, $t[x_1.v_1\xi, x_2.v_2\xi](\sigma_i\{v/s\})$ has the same complexity as $(t[x_1.v_1, x_2.v_2]\xi)\sigma_i$.

- ${}_a(t_1 \parallel \ldots \parallel t_m)\{v/s\}$. By the induction hypothesis, $(*)$ holds for $t_i\{v/s\}$ where $1 \le i \le m$. The only redex in ${}_a(t_1 \parallel \ldots \parallel t_m)\{v/s\}$ and not in some $t_i\{v/s\}$ can be ${}_a(t_1\{v/s\} \parallel \ldots \parallel t_m\{v/s\})$ itself. But the complexity of such a redex equals the maximal complexity of the occurrences of the channel a in $t_i\{v/s\}$. Hence the statement follows.

□

The following result is meant to guarantee that there cannot be any loop involving intuitionistic redexes and communication redexes of non-decreasing complexity. Intuitively, when Group 1 redexes generate new redexes, the latter have smaller complexity than the former; when Group 2 redexes generate new redexes, the latter do not have greater complexity than the former. Hence, during the communication phase the complexity does not increase and during the intuitionistic phase the complexity strictly decreases.

Proposition 4.9 (Decrease!). *Let t be a term in parallel form, r be one of its redexes of complexity τ, and t' be the term that we obtain from t by reducing r.*
1. *If r is a redex of Group 1, then the complexity of each redex in t' is not greater than the complexity of a redex of the same group occurring in t; or not greater than the complexity of a case distinction permutation redex occurring in t; or smaller than τ.*
2. *If r is a redex of Group 2 and not an activation, then the complexity of any redex in t' is either equal to the complexity of a redex of the same group in t or not greater than τ.*

Proof.
1. Suppose that $r = (\lambda x^A\, s)\, v$, that $s : B$ and let q be a redex in t' whose complexity is different from the complexity of any redex of the same group and any case distinction permutation occurring in t. We apply Lemma 4.6 to the term $s[v/x^A]$. From such lemma we can infer that if q occurs in $s[v/x^A]$, since (i) and (ii) do not apply, it has the same value complexity as v, which by Proposition 4.1 is at most the complexity of A, and hence strictly smaller than the complexity of $A \to B$ and than the complexity τ of r. Assume therefore that q does not occur in $s[v/x^A]$. Since $s[v/x^A] : B$, by applying Lemma 4.6 to the term $t' = t\{s[v/x^A]/(\lambda x^A\, s)\, v\}$ we know that either q has the same complexity as the value complexity of $s[v/x^A]$ – which by Proposition 4.1 is at most the complexity of B – or q is a communication redex of complexity equal to the complexity of some channel occurrence $a(w[v/x^A])$ in $s[v/x^A]$, which by Lemma 4.5 is either at most the complexity of A or at most the complexity of aw.

Suppose that $r = \iota_i(s)[x^A.u_1, y^B.u_2]$. By applying Lemma 4.7 to

$$t' = t\{u_i[s/x_i^{A_i}]\,/\,\iota_i(s)[x_1^{A_1}.u_1, x_2^{A_2}.u_2]\}$$

we are done.

2. Suppose that $r = \langle v_0, v_1\rangle \pi_i$ for $i \in \{0,1\}$, that $\langle v_0, v_1\rangle : A_0 \wedge A_1$ and let q be a redex in t' having complexity greater than that of any redex of the same group in t. Then the assumption just made implies that the term q cannot occur in v_i, but it must have been created by the reduction of r. Moreover, by Proposition 4.1, the value complexity of v_i cannot be greater than the complexity of A_i. By applying Lemma 4.6 to the term $t' = t\{v_i/r\}$, we know that q has complexity equal to the value complexity of v_i, because by the assumption on q the cases (i) and (ii) of Lemma 4.6 do not apply. Such complexity is at most the complexity τ of r. Thus we are done.

Suppose that $r = s[x^A.u, y^B.v]\xi$ is a case distinction permutation redex. By applying Lemma 4.8 we are done.

If t' is obtained by performing a permutation of a parallelism operator, then obviously the thesis holds.

If t' is obtained by a simplification reduction of the form ${}_a(u_1 \parallel \dots \parallel u_m) \mapsto u_{j_1} \parallel \dots \parallel u_{j_n}$, for $1 \leq j_1 < \dots < j_n \leq m$, then there is nothing to prove: all redexes occurring in t' also occur in t.

Suppose now that

$$r = {}_a(\mathcal{C}_1[a^{F_1 \to G_1}\, t_1] \parallel \dots \parallel \mathcal{C}_m[a^{F_m \to G_m}\, t_m])$$

$t_i = \langle u^i_1, \dots, u^i_{p_i}\rangle$. Then by cross reduction, r reduces to ${}_b(s_1 \parallel \dots \parallel s_m)$ where if $G_i \neq \bot$:

$$s_i \;=\; {}_a(\mathcal{C}_1[a^{F_1 \to G_1}\, t_1] \dots \parallel \mathcal{C}_i[t_j^{b^{B_i \to B_j}\langle \boldsymbol{y}_i\rangle/\boldsymbol{y}_j}] \parallel \dots \parallel \mathcal{C}_m[a^{F_m \to G_m}\, t_m])$$

and if $G_i = \bot$:

$$s_i \;=\; {}_a(\mathcal{C}_1[a^{G_1 \to G_1}\, t_1] \dots \parallel \mathcal{C}_i[b^{B_i \to \bot}\langle \boldsymbol{y}_i\rangle] \parallel \dots \parallel \mathcal{C}_m[a^{F_m \to G_m}\, t_m])$$

in which $F_j = G_i$. and $t' = t\{r'/r\}$. Let now q be a redex in t' having different complexity from the one of any redex of the same group in t. We first show that it cannot be an intuitionistic redex: assume it is. Then it occurs in one of the terms $\mathcal{C}_i[t_j^{b^{B_i \to B_j}\langle \boldsymbol{y}_i\rangle/\boldsymbol{y}_j}]$ or $\mathcal{C}_i[b^{B_i \to \bot}\langle \boldsymbol{y}_i\rangle]$. By applying Lemma 4.6 to them, we obtain that the complexity of q is the value complexity of $t_j^{b^{B_i \to B_j}\langle \boldsymbol{y}_i\rangle/\boldsymbol{y}_j}$ or $b^{B_i \to \bot}\langle \boldsymbol{y}_i\rangle$, which, by several applications of Lemma 4.5, are at most the value complexity of t_j or 0 and thus by definition at most the complexity of r, which is a contradiction. Assume hence that $q = {}_c(p_1 \parallel \dots \parallel p_z)$ is a communication redex. Every channel occurrence of c in the terms $\mathcal{C}_i[t_j^{b^{B_i \to B_j}\langle \boldsymbol{y}_i\rangle/\boldsymbol{y}_j}]$ or $\mathcal{C}_i[b^{B_i \to \bot}\langle \boldsymbol{y}_i\rangle]$ is of the form $cw\{t_j^{b^{B_i \to B_j}\langle \boldsymbol{y}_i\rangle/\boldsymbol{y}_j}/a\, t_i\}$ or $cw\{b^{B_i \to \bot}\langle \boldsymbol{y}_i\rangle/a\, t_i\}$ where cw is a channel occurrence in t. By Lemma 4.5, each of these occurrences has either at most the value complexity of cw or has at most the value complexity of r, which is again a contradiction. □

The following result is meant to rule out loops between communication and activation redexes: no new activation is generated after a cross reduction if there is none to start with.

Lemma 4.10 (Freeze!). *Suppose that s is a term in parallel form that does not contain projection, case distinction permutation or activation redexes. Let ${}_a(q_1 \parallel \dots \parallel q_m)$ be some redex in s of complexity τ. If s' is obtained from s by performing first a cross reduction on ${}_a(q_1 \parallel \dots \parallel q_m)$ and then reducing all projection and case distinction permutation redexes, then s' contains no activation redexes.*

Proof. Let ${}_a(q_1 \parallel \dots \parallel q_m) = {}_a(\mathcal{C}_1[a^{F_1\to G_1}\, t_1\sigma_1] \parallel \dots \parallel \mathcal{C}_m[a^{F_m\to G_m}\, t_m\sigma_m])$ where σ_i for $1 \leq i \leq m$ are the stacks which are applied to $a\, t_i$, and t be the cross reduction redex occurring in s that we reduce to obtain s', or in other terms $s' = s\{t'/t\}$. Then after performing the cross reduction and reducing all the intuitionistic redexes, t reduces to ${}_b(s_1 \parallel \dots \parallel s_m)$ where if $G_i \neq \bot$, then

$$s_i \;=\; {}_a(\mathcal{C}_1[a^{F_1\to G_1}\, t_1] \dots \parallel \mathcal{C}_i[t'_j] \parallel \dots \parallel \mathcal{C}_m[a^{F_m\to G_m}\, t_m])$$

and if $G_i = \bot$, then

$$s_i \;=\; {}_a(\mathcal{C}_1[a^{G_1\to G_1}\, t_1] \dots \parallel \mathcal{C}_i[b^{B_i\to\bot}\langle \boldsymbol{y}_i\rangle] \parallel \dots \parallel \mathcal{C}_m[a^{F_m\to G_m}\, t_m])$$

in which $F_j = G_i$ and t'_j are the terms obtained reducing all projection and case distinction permutation redexes in $t_j^{b^{B_i\to B_j}\langle \boldsymbol{y}_i\rangle/\boldsymbol{y}_j}$.

We observe that a is active and hence the terms s_i for $1 \leq i \leq m$ are not activation redexes. Moreover, since all occurrences of b are of the form $b^{B_i\to\bot}\langle \boldsymbol{y}_i\rangle$, t' is not an activation redex. Now we consider the channel occurrences in any term s_i. We first show that there are no activable channels in t'_j which are bound in s'. Since for any stack θ of projections $b\langle \boldsymbol{y}_i\rangle\theta$ has value complexity 0, for any subterm w of t_j we can apply repeatedly Lemma 4.5 to $w^{b^{B_i\to B_j}\langle \boldsymbol{y}_i\rangle/\boldsymbol{y}_j}$ and obtain that the value complexity of $w^{b^{B_i\to B_j}\langle \boldsymbol{y}_i\rangle/\boldsymbol{y}_j}$ is exactly the value complexity of w. This implies that there is no activable channel in $t_j^{b^{B_i\to B_j}\langle \boldsymbol{y}_i\rangle/\boldsymbol{y}_j}$ because there was none in t_j. The following general statement immediately implies that there is no activable channel in t'_j which is bound in s' either.

$(*)$ Suppose that r and θ are respectively a simply typed λ-term and a stack contained in s' that do not contain projection and permutation redexes, nor activable channels bound in s'. If r' is obtained from $r\theta$ by performing all possible projection and case distinction permutation reductions, then there are no activable channels in r' which are bound in s'.

We prove $(*)$ by induction on the size of r and proceed by cases according to its shape.

- If $r = \lambda x w$, $r = \iota_i(w)$, $r = w\,\mathsf{efq}_P$, $r = x$ or $r = aw$, for a channel a, then $r' = r\theta$ and the thesis holds.

- If $r = \langle v_0, v_1\rangle$ the only redex that can occur in $r\theta$ is a projection redex, when $\theta = \pi_i\rho$. Hence, $r\theta \mapsto v_i\rho \mapsto^* r'$. Since v_i cannot be a channel because otherwise $\langle v_0, v_1\rangle$ would not be a legal term, we can apply the induction hypothesis on v_i and ρ. Hence, there are no activable channels in r' which are bound in s'.

- If $r = t[x.v_0, y.v_1]$, then $r\theta \mapsto^* t[x.v_0\theta, y.v_1\theta] \mapsto^* t[x.v'_0, y.v'_1] = r'$. By the induction hypothesis applied to v_0 and θ and to v_1 and θ, there are no activable channels in v'_0 and v'_1 and we are done.

- If $r = p\nu\xi$, where ξ is a case free stack of length 1, then $r' = p\nu\xi$ and the thesis holds.

Now, let c be any non-active channel bound in s' occurring in $\mathcal{C}_i[t'_j]$ or $\mathcal{C}_i[b^{B_i\to\bot}\langle \boldsymbol{y}_i\rangle]$ but not in u': any of its occurrences is of the form $c\langle p_1, \dots, p_l\{u'/av\rho\}, \dots, p_n\rangle$, where each p_l is not a pair. We want to show that the value complexity of $p_l\{u'/av\rho\}$ is exactly the value complexity of p_l. Indeed, $p_l = r\,\nu$ where ν is a case-free stack. If r is of the form $\lambda x w$, $\langle q_1, q_2\rangle$, $\iota_i(w)$, x, dw, with $d \neq a$, then the value complexity of $p_l[u'/av\rho]$ is the same as that of p_l; note, indeed, that if $r = \langle q_1, q_1\rangle$, then ν is not empty because $r\,\nu$ is assumed not to be a pair. If $r = v_0[x_1.v_1, x_2.v_2]$, then ν is empty, otherwise s would contain a permutation redex, so $c\langle p_1, \dots, p_l, \dots, p_n\rangle$ is activable, and there is an activation redex in s, which is contrary to our assumptions. The case $r = av$ and $\nu = \rho\rho'$ is also impossible, otherwise $c\langle p_1, \dots, p_l, \dots, p_n\rangle$ would be activable, and we are done. $\square$

Definition 4.10. The height of a term t in parallel form is

- 0 if t is a simply typed λ-term
- $1 + \max(m, n)$ if $t = u \parallel_a v$ and the heights of u and v are m and n respectively.

We show that the communication phase of our reduction strategy is finite.

Lemma 4.11 (Communicate!). *Let t be any term in parallel form that does not contain projection, case distinction permutation, or activation redexes. Assume moreover that all redexes in t have complexity at most τ. Then t reduces to a term containing no redexes, except Group 1 redexes of complexity at most τ.*

Proof. We prove the statement by lexicographic induction on the triple (n, h, g) where

- n is the number of subterms $_a(u_1 \parallel \ldots \parallel u_m)$ of t such that $_a(u_1 \parallel \ldots \parallel u_m)$ is an active, but not uppermost, session.
- h is the function mapping each natural number $m \geq 2$ into the number of uppermost active sessions in t with height m.
- g is the function mapping each natural number m into the number of uppermost active sessions $_a(u_1 \parallel \ldots \parallel u_m)$ in t containing m occurrences of a.

We employ the following lexicographic ordering between functions for the second and third elements of the triple: $f < f'$ if and only if there is some i such that for all $j > i$, $f(j) = f'(j) = 0$ and $f(i) < f'(i)$.

If $h(j) > 0$, for some $j \geq 2$, then there is at least an active session $_a(u_1 \parallel \ldots \parallel u_m)$ in t that does not contain any active session and such that the height of $_a(u_1 \parallel \ldots \parallel u_m)$ is j. Hence $u_i = {}_b(s_1 \parallel \ldots \parallel s_q)$ for some $1 \leq i \leq j \leq m$. We obtain t' by applying inside t the permutation $_a(u_1 \parallel \ldots \parallel {}_b(s_1 \parallel \ldots \parallel s_q) \ldots \parallel u_m) \mapsto {}_b({}_a(u_1 \parallel \ldots \parallel s_1 \ldots \parallel u_m) \ldots \parallel {}_a(u_1 \parallel \ldots \parallel s_q \ldots \parallel u_m))$. We claim that the term t' thus obtained has complexity (n, h', g'), with $h' < h$. Indeed, $_a(u_1 \parallel \ldots \parallel u_m)$ does not contain active sessions, thus b is not active and the number of active sessions which are not uppermost in t' is still n. With respect to t, the term t' contains one less uppermost active session with height j and q more of height $j - 1$, and hence $h' < h$. Furthermore, since the permutations do not change at all the purely intuitionistic subterms of t, no new activation or intuitionistic redex is created. In conclusion, we can apply the induction hypothesis on t' and thus obtain the thesis.

If $h(m) = 0$ for all $m \geq 2$, then let us consider an uppermost active session $_a(u_1 \parallel \ldots \parallel u_m)$ in t such that the height of $_a(u_1 \parallel \ldots \parallel u_m)$ is 1; if there is none, we are done. We reason by cases on the distribution of the occurrences of a. Either (i) some u_i for $1 \leq i \leq m$ does not contain any occurrence of a, or (ii) all u_i for $1 \leq i \leq m$ contain some occurrence of a.

Suppose that (i) is the case and, without loss of generality, that a occurs j times in u and does not occur in v. We then obtain a term t' by applying a simplification reduction $_a(u_1 \parallel \ldots \parallel u_m) \mapsto u_{j_1} \parallel \ldots \parallel u_{j_p}$. If there is an active session $_b(s_1 \parallel \ldots \parallel s_q)$ in t such that $_a(u_1 \parallel \ldots \parallel u_m)$ is the only active session contained in some s_i for $1 \leq i \leq n$, then the term t' has complexity $(n - 1, h', g')$, because $_b(s_1 \parallel \ldots \parallel s_q)$ is an active session which is not uppermost in t, but is uppermost in t'; if not, we claim that the term t' has complexity (n, h, g') where $g' < g$. Indeed, first, the number of active sessions which are not uppermost does not change. Second, the height of all other uppermost active sessions does not change. Third, $g'(j) = g(j) - 1$ and, for any $i \neq j$, $g'(i) = g(i)$ because, obviously, no channel belonging to any uppermost active session different from $_a(u_1 \parallel \ldots \parallel u_m)$ occurs in u. Since the reduction

${}_a(u_1 \parallel \dots \parallel u_m) \mapsto u_{j_1} \parallel \dots \parallel u_{j_p}$ does not introduce any new intuitionistic or activation redex, we can apply the induction hypothesis on t' and obtain the thesis.

Suppose now that (ii) is the case and that all u_i for $1 \leq i \leq m$ together contain j occurrences of a. Then ${}_a(u_1 \parallel \dots \parallel u_m)$ is of the form ${}_a(\mathcal{C}_1[a^{F_1 \to G_1}\, t_1] \parallel \dots \parallel \mathcal{C}_m[a^{F_m \to G_m}\, t_m])$ where a is active, $\mathcal{C}_j[a^{F_j \to G_j}\, t_j]$ for $1 \leq j \leq m$ are simply typed λ-terms; $a^{F_j \to G_j}$ is rightmost in each of them. Then we can apply the cross reduction ${}_a(\mathcal{C}_1[a^{F_1 \to G_1}\, t_1] \parallel \dots \parallel \mathcal{C}_m[a^{F_m \to G_m}\, t_m]) \mapsto {}_b(s_1 \parallel \dots \parallel s_m)$ in which b is fresh and for $1 \leq i \leq m$, we define, if $G_i \neq \bot$, then

$$s_i \;=\; {}_a(\mathcal{C}_1[a^{F_1 \to G_1}\, t_1] \dots \parallel \mathcal{C}_i[t_j^{\,b^{B_i \to B_j} \langle \boldsymbol{y}_i \rangle / \boldsymbol{y}_j}] \parallel \dots \parallel \mathcal{C}_m[a^{F_m \to G_m}\, t_m])$$

and if $G_i = \bot$, then

$$s_i \;=\; {}_a(\mathcal{C}_1[a^{G_1 \to G_1}\, t_1] \dots \parallel \mathcal{C}_i[b^{B_i \to \bot} \langle \boldsymbol{y}_i \rangle] \parallel \dots \parallel \mathcal{C}_m[a^{F_m \to G_m}\, t_m])$$

where $F_j = G_i$; $\boldsymbol{y}_z$ for $1 \leq z \leq m$ is the sequence of the free variables of t_z bound in $\mathcal{C}_z[a^{F_z \to G_z}\, t_z]$; B_z for $1 \leq z \leq m$ is the type of $\langle \boldsymbol{y}_z \rangle$. By Lemma 4.10, after performing all projections and case permutation reductions in all $\mathcal{C}_i[t_j^{\,b^{B_i \to B_j} \langle \boldsymbol{y}_i \rangle / \boldsymbol{y}_j}]$ and $\mathcal{C}_i[b^{B_i \to \bot} \langle \boldsymbol{y}_i \rangle]$ for $1 \leq i \leq m$ we obtain a term t' that contains no activation redexes; moreover, by point 2. of Proposition 4.9, t' contains only redexes having complexity at most τ.

We claim that the term t' thus obtained has complexity $\langle n, h, g' \rangle$ where $g' < g$. Indeed, the value n does not change because all newly introduced occurrences of b are not active. The new active sessions $s_i = {}_a(\mathcal{C}_1[a^{F_1 \to G_1}\, t_1] \dots \parallel \mathcal{C}_i[t_j^{\,b^{B_i \to B_j} \langle \boldsymbol{y}_i \rangle / \boldsymbol{y}_j}] \parallel \dots \parallel \mathcal{C}_m[a^{F_m \to G_m}\, t_m])$ or $s_i = {}_a(\mathcal{C}_1[a^{G_1 \to G_1}\, t_1] \dots \parallel \mathcal{C}_i[b^{B_i \to \bot} \langle \boldsymbol{y}_i \rangle] \parallel \dots \parallel \mathcal{C}_m[a^{F_m \to G_m}\, t_m])$ for $1 \leq i \leq m$ all have height 1 and contain $j - 1$ occurrences of a. Since furthermore the reduced term does not contain channel occurrences of any uppermost active session different from ${}_a(u_1 \parallel \dots \parallel u_m)$ we can infer that $g'(j) = g(j) - 1$ and that, for any i such that $i > j$, $g'(i) = g(i)$.

We can apply the induction hypothesis on t' and obtain the thesis. □

We combine the main results achieved so far to prove that the normalization procedure, applied to a proof term in parallel form, terminates and yields a term in normal form.

Proposition 4.12 (Normalize!). *Let $t : A$ be any term in parallel form. Then $t \mapsto^* t'$, where t' is in parallel normal form.*

Proof. By induction on the maximum complexity τ of redexes in t. Starting from t, we reduce all intuitionistic redexes and obtain a term t_1 that, by Proposition 4.9, does not contain redexes of complexity greater than τ. By Lemma 4.3, $t_1 \mapsto^* t_2$ where t_2 does not contain any redex, except cross reduction redexes of complexity at most τ. By Lemma 4.11, $t_2 \mapsto^* t_3$ where t_3 contains only Group 1 redexes of complexity at most τ. Suppose $t_3 \mapsto^* t_4$ by reducing all Group 1 redexes, starting from t_3. By Proposition 4.9, every Group 1 redex generated in the process has complexity at most τ, thus every Group 2 redex which is generated has complexity smaller than τ, thus t_4 can only contain redexes with complexity smaller than τ. By the induction hypothesis $t_4 \mapsto^* t'$, with t' in parallel normal form. □

The normalization of all λ_{L} proof terms easily follows.

Theorem 4.13 (Normalization Theorem). *Suppose that $t : A$ is a λ_{L} proof term. Then $t \mapsto^* t' : A$, where t' is a normal parallel form.*

Proof. By Proposition 4.2 and 4.12. □

5 The Subformula Property

We prove that normal λ_{L}-terms satisfy the subformula property: they represent proofs only containing concepts that already appear in their premises or conclusion. Hence, the Curry–Howard correspondence for λ_{L} is meaningful from a logical perspective.

We first show that every normal λ_{L}-term is in parallel form.

Proposition 5.1 (Parallel Form Property)**.** *If $t \in \mathrm{NF}$ is a λ_{L}-term, then it is in parallel form.*

Proof. By induction on the structure of t. See A.5, in the appendix, for the proof. □

For the subformula property, we shall need the following definition.

Definition 5.1 (Prime Formulas and Factors [26])**.** A formula is said to be **prime** if it is not a conjunction. Every formula is a conjunction of prime formulas, called **prime factors**.

Theorem 5.2 (Subformula Property)**.** *If $x_1^{A_1}, \dots, x_n^{A_n}, a_1^{D_1}, \dots, a_m^{D_m} \vdash t : A$ and $t \in \mathrm{NF}$, then:*

1. *For each channel variable $a^{B \to C}$ occurring bound in t, the prime factors of B, C are subformulas of $A_1, \dots, A_n, A$ or proper subformulas of $D_1, \dots, D_m$.*
2. *The type of any subterm of t is either a subformula or a conjunction of subformulas of $A_1, \dots, A_n, A$ and of proper subformulas of $D_1, \dots, D_m$.*

Proof. By structural induction on t and reasoning on the form of t.

- $t = \langle u, v\rangle : F \wedge G$. Since $t \in \mathrm{NF}$, by Proposition 5.1 it is in parallel form, thus is a simply typed λ-term. Hence no communication variable can be bound inside t, thus 1. trivially holds. By the induction hypothesis, 2. holds for $u : F$ and $v : G$. Hence, the type of any subterm of u is either a subformula or a conjunction of subformulas of $A_1, \dots, A_n$, of F and of proper subformulas of $D_1, \dots, D_m$ and any subterm of v is either a subformula or a conjunction of subformulas of some $A_1, \dots, A_n$, of G and of proper subformulas of $D_1, \dots, D_m$. Moreover, any subformula of F and G must be a subformula of the type $F \wedge G$ of t. Hence the type of any subterm of $\langle u, v\rangle$ is either a subformula or a conjunction of subformulas of $A_1, \dots, A_n, F \wedge G$ or a proper subformula of $D_1, \dots, D_m$ and the statement holds for t as well.
- $t = \lambda x^F\, u : F \to G$. Since $t \in \mathrm{NF}$, by Proposition 5.1 it is in parallel form, thus is a simply typed λ-term. Hence no communication variable can be bound inside t, thus 1. trivially holds. By the induction hypothesis, 2. holds for $u : G$. Hence the type of any subterm of u is either a subformula or a conjunction of subformulas of some $A_1, \dots, A_n, F$, of G and of proper subformulas of $D_1, \dots, D_m$. Since the type F of x is a subformula of $F \to G$, the type of any subterm of $\lambda x^F\, u$ is either a subformula or a conjunction of subformulas of $A_1, \dots, A_n, F \to G$ or a proper subformula of $D_1, \dots, D_m$ and the statement holds for t as well.
- $t = \iota_i(u) : F \vee G$ for $i \in \{0, 1\}$. Without loss of generality assume that $i = 1$ and $u : G$. Since $t \in \mathrm{NF}$, by Proposition 5.1 it is in parallel form, thus is a simply typed λ-term. Hence no communication variable can be bound inside t, thus 1. trivially holds. By the induction hypothesis, 2. holds for $u : F$. Hence, the type of any subterm of u is either a subformula or a conjunction of subformulas of some $A_1, \dots, A_n$, of F or proper subformulas of $D_1, \dots, D_m$. Moreover, any subformula of G must be a subformula of the type $F \vee G$ of t. Hence the type of any subterm of $\iota_i(u)$ is either a subformula or a conjunction of subformulas of $A_1, \dots, A_n, F \vee G$ or a proper subformula of $D_1, \dots, D_m$ and the statement holds for t as well.

- $t = x^{A_i}\,\sigma : A$ for some A_i among $A_1, \ldots, A_n$ and stack σ. Since $t \in \mathrm{NF}$, it is in parallel form, thus is a simply typed λ-term and no communication variable can be bound inside t. By the induction hypothesis, for any element $\sigma_j : S_j$ of σ, the type of any subterm of σ_j is either a subformula or a conjunction of subformulas of some $A_1, \ldots, A_n$, of the type S_j of σ_j and of proper subformulas of $D_1, \ldots, D_m$.

  If σ is case-free, then every S_j is a subformula of A_i, or of A, when $\sigma = \sigma' \mathsf{efq}_A$. Hence, the type of any subterm of $x^{A_i}\,\sigma$ is either a subformula or a conjunction of subformulas of $A_1, \ldots, A_n, A$ or of proper subformulas of $D_1, \ldots, D_m$ and the statement holds for t as well.

  In case σ is not case-free, then, because of case distinction permutations, $\sigma = \sigma'[y^G.v_1, z^E.v_2]$, with σ' case-free. By the induction hypothesis we know that the type of any subterm of $v_1 : A$ or $v_2 : A$ is either a subformula or a conjunction of subformulas of some $A_1, \ldots, A_n$, of A, G, E and of proper subformulas of $D_1, \ldots, D_m$. Moreover, G and E are subformulas of A_i due to the properties of stacks. Hence, the type of any subterm of $x\,\sigma'[y^G.v_1, z^E.v_2]$ is either a subformula or a conjunction of subformulas of $A_1, \ldots, A_n, A$ and of proper subformulas of $D_1, \ldots, D_m$ and also in this case the statement holds for t as well.

- $t = a^{D_i} u\,\sigma : A$ for some D_i among $D_1, \ldots, D_n$ and stack σ. As in the previous case.

- $t = {}_b(u_1 \parallel \ldots \parallel u_k) : A$ and $b^{G_i \to H_i}$ occurs in u_i. Suppose, for the sake of contradiction, that the statement does not hold. We know by the induction hypothesis that the statement holds for $u_1 : A, \ldots, u_k : A$. We first show that it cannot be the case that

  > $(*)$ all prime factors of $G_1, H_1, \ldots, G_k, H_k$ are subformulas of $A_1, \ldots, A_n, A$ or proper subformulas of $D_1, \ldots, D_m$.

  Indeed, assume by contradiction that $(*)$ holds. Let us consider the type T of any subterm of t which is not a bound communication variable and the formulas B, C of any bound communication variable $a^{B \to C}$ of t. Let P be any prime factor of T or B or C. By the induction hypothesis applied to $u_1, \ldots, u_n$, we obtain that P is either subformula or conjunction of subformulas of $A_1, \ldots, A_n, A$ and of proper subformulas of $D_1, \ldots, D_m, G_1, H_1, \ldots, G_k, H_k$. Moreover, P is prime and so it must be subformula of $A_1, \ldots, A_n, A$ or a proper subformula of $D_1, \ldots, D_m$ or a prime factor of $G_1, H_1, \ldots, G_k, H_k$. Since $(*)$ holds, P must be a subformula of $A_1, \ldots, A_n, A$ or proper subformula of $D_1, \ldots, D_m$, and this contradicts the assumption that the subformula property does not hold for t.

  We shall say from now on that any bound channel variable $a^{F_1 \to F_2}$ of t *violates the subformula property maximally (due to Q)* if (i) some prime factor Q of F_1 or F_2 is neither a subformula of $A_1, \ldots, A_n, A$ nor a proper subformula of $D_1, \ldots, D_m$ and (ii) for every other bound channel variable $c^{S_1 \to S_2}$ of t, if some prime factor Q' of S_1 or S_2 is neither a subformula of $A_1, \ldots, A_n, A$ nor a proper subformula of $D_1, \ldots, D_m$, then Q' is complex at most as Q. If Q is a subformula of F_1 we say that $a^{F_1 \to F_2}$ violates the subformula property maximally *in the input.*

  It follows from $(*)$ that a channel variable maximally violating the subformula property must exist. We show now that there also exists a subterm $c^{F_1 \to F_2} w$ of t such that c maximally violates the subformula property in the input due to Q, and w does not contain any channel variable that violates the subformula property maximally.

  In order to prove the existence of such term, we prove

  $(**)$ Let t_1 be any subterm of t such that

– t_1 contains at least a maximally violating channel of t

– all maximally violating channels of t that are free in t_1 are maximally violating in the input.

Then there is a simply typed subterm s of t_1 such that s contains at least a maximally violating channel of t, and such that all occurrences of maximally violating channels of t occurring in s violate the subformula property in the input.

Intuitively, in order to find such a simply typed λ-term s we consider by induction the immediate subterms of t_1. If we find a subterm $_{d}(v_1 \parallel \ldots \parallel v_n)$ such that d is not maximally violating, we continue the search in the subterms $v_1, \ldots, v_n$. If we find a subterm $_{d}(v_1 \parallel \ldots \parallel v_n)$ such that d is maximally violating, we continue the search in the subterm v_i for $i \in \{1, \ldots, n\}$ that contains the occurrences of d which are maximally violating in the input. Formally, we proceed by induction on the number n of $\parallel$ operators that occur in t_1.channels.

If $n = 0$, it is enough to pick $s = t_1$.

If $n > 0$, let $t_1 = {}_{d}(v_1 \parallel \ldots \parallel v_n)$ and assume $d^{E_i \to F_i}$ occurs in v_i. If no $d^{E_i \to F_i}$ maximally violates the subformula property, we obtain the thesis by applying the induction hypothesis to the terms v_i. Assume hence that some $d^{E_i \to F_i}$ maximally violates the subformula property due to Q. Then there is some $d^{E_j \to F_j}$ such that Q is a prime factor of E_i or E_j. By the induction hypothesis applied to v_i or v_j, we obtain the thesis.

By $(**)$ we can infer that in t there is a simply typed λ-term s that contains at least one occurrence of a maximally violating channel of t and only occurrences of maximally violating channels of t that are maximally violating in the input. Which means that the rightmost of the maximally violating channel occurrences in s is of the form $c^{F_1 \to F_2} w$ where c maximally violates the subformula property in the input and w does not contain any channel variable maximally violating the subformula property.

Consider now the term $c^{F_1 \to F_2} w$. Since Q is a prime factor of F_1, it is either an atom P or a formula of the form $Q' \to Q''$ or of the form $Q' \vee Q''$. Let $w = \langle w_1, \ldots, w_j \rangle$, where each w_i is not a pair, and let k be such that Q occurs in the type of w_k.

We start by ruling out the case that $w_k = \lambda y\, s$ or $w_k = \iota_i(s)$ for $i \in \{0, 1\}$, otherwise, since w would be a value, it would be possible to perform an activation reduction or a cross reduction to some subterm $_{c}(u'_1 \parallel \ldots \parallel u'_{m'})$, which must exist since c is bound.

Suppose now, by contradiction, that $w_k = x^T\, \sigma$ where σ is a stack. It cannot be the case that $\sigma = \sigma'[y^{E_1}.v_1, z^{E_2}.v_2]$ or that $\sigma = \sigma' \mathsf{efq}_P$, because otherwise we could apply an activation reduction or a cross reduction. Hence σ is case-free and does not contain efq_P. Moreover, x^T cannot be a free variable of t, because then T would be equal to some A_i for $1 \leq i \leq n$, and Q would be a subformula of A_i, which contradicts the assumptions. Suppose hence that x^T is a bound intuitionistic variable of t, such that t has a subterm $\lambda x^T s : T \to Y$ or, without loss of generality, $s[x^T.v_1, z^E.v_2]$, with $s : T \vee Y$ for some formula Y. By the main induction hypothesis, $T \to Y$ and $T \vee Y$ are subformulas of $A_1, \ldots, A_n, A$ or proper subformulas of $D_1, \ldots, D_m, G \to H$. But $T \to Y$ and $T \vee Y$ contain Q as a proper subformula and $c^{F_1 \to F_2}\, w$ violates maximally the subformula due to Q. Hence $T \to Y$ and $T \vee Y$ are neither subformulas of $A_1, \ldots, A_n, A$ nor proper subformulas of $D_1, \ldots, D_m$ and thus must be proper subformulas of $G \to H$. Since $c^{F_1 \to F_2}\, w$ violates the subformula property maximally due to Q, $T \to Y$ and $T \vee Y$ must be at most as complex as Q, which is a contradiction. Suppose now that x^T is a bound channel variable, thus $w_k = a^T\, r\, \sigma$, where a^T is a bound communication variable of t, with $T = T_1 \to T_2$. Since

$c^{F_1 \to F_2}\, w$ is rightmost, $a \neq c$. Moreover, Q is a subformula of a prime factor of T_2, whereas $a^{T_1 \to T_2}$ occurs in w, which is impossible by choice of c. This contradicts the assumption that the term is normal and ends the proof.

□

6 Conclusions and Open Problems

We defined Curry–Howard correspondences for a large class of intermediate logics characterized by cut-free hypersequent calculi. These logics include well-known systems such as classical and Gödel–Dummett logic. The correspondences give rise to typed concurrent λ-calculi, each of which features specific communication mechanisms; moreover the reductions needed to obtain the subformula property (*cross reductions*) have a natural interpretation in terms of code mobility. This confirms Avron's intuition on the computational content of the intermediate logics formalized as hypersequent calculi [5].

Although the paper's results are conclusive in this respect, they open new research lines. The first one concerns the extension of our correspondences to first-order logics. Even though the research direction is very natural, already in the case of hypersequents, the definition of suitable quantifier rules is not straightforward. Hypersequent rules for quantifiers were first introduced in [7] for Gödel logic, also known as Intuitionistic Fuzzy logic. The eigenvariable condition for them, which became the standard for first-order hypersequent calculi, hinders the translation of hypersequent rules into systems of rules [15]. This condition also interferes with the transformation of proofs into *parallel form* – see Proposition 4.2 – which is essential for the normalization procedure used in the present work. Thus, new ideas are required to define normalizing natural deduction calculi with a natural computational interpretation for first-order intermediate logics.

A problem of concurrency theory which has remained open for several decades is the definition of a formalism providing a foundation for concurrent functional programs. For sequential computation, such a foundation is provided by λ-calculus. According to Milner, the introduction of CCS itself – the forefather of process calculi – was due to the unsuccessful attempts to define satisfactory concurrent extensions of λ-calculus [29]. While the definition of such an untyped concurrent λ-calculus lies outside the scope of this work, the set of reductions presented for λ_{L} might shed light on possible approaches to the solution of this problem.

References

[1] S. Abramsky. Computational interpretations of linear logic. *Theor. Comput. Sci.*, 111(1-2):3–57, 1993.

[2] F. Aschieri, A. Ciabattoni, and F.A. Genco. Gödel logic: from natural deduction to parallel computation. In *LICS 2017*, pages 1–12, 2017.

[3] F. Aschieri, A. Ciabattoni, and F.A. Genco. Classical proofs as parallel programs. In *GandALF 2018*, pages 43–57, 2018.

[4] A. Avron. A constructive analysis of RM. *The Journal of Symbolic Logic*, 52(4):939–951, 1987.

[5] A. Avron. Hypersequents, logical consequence and intermediate logics for concurrency. *Annals of Mathematics and Artificial Intelligence*, 4(3):225–248, 1991.

[6] M. Baaz, A. Ciabattoni, and C. Fermüller. A natural deduction system for intuitionistic fuzzy logic. In *Lectures on Soft Computing and Fuzzy Logic*, pages 1–18. Physica-Verlag, 2000.

[7] M. Baaz and R. Zach. Hypersequents and the proof theory of intuitionistic fuzzy logic. In *CSL'00. LNCS*, pages 187–201. Springer, 2000.

[8] A. Beckmann and N. Preining. Hyper natural deduction. In *LICS 2015*, pages 547–558, 2015.

[9] A. Beckmann and N. Preining. Hyper natural deduction for Gödel logic a natural deduction system for parallel reasoning. *J. of Logic and Computation*, 28(6):1125–1187, 2018.

[10] G. Boudol. Towards a lambda-calculus for concurrent and communicating systems. In *TAPSOFT 1998*, pages 149–161, 1989.

[11] L. Caires and F. Pfenning. Session types as intuitionistic linear propositions. In *CONCUR 2010*, pages 222–236, 2010.

[12] M. Carbone, F. Montesi, and C. Schürmann. Choreographies, logically. *Distributed Computing*, 31(1):51–67, 2018.

[13] L. Cardelli and A.D. Gordon. Mobile ambients. *Theor. Comput. Sci.*, 240(1):177–213, 2000.

[14] A. Ciabattoni, N. Galatos, and K. Terui. From axioms to analytic rules in nonclassical logics. In *LICS 2008*, pages 229–240, 2008.

[15] A. Ciabattoni and F.A. Genco. Hypersequents and systems of rules: Embeddings and applications. *TOCL*, 19(2):11:1–11:27, 2018.

[16] V. Danos and J.-L. Krivine. Disjunctive tautologies as synchronisation schemes. *CSL 2000*, 1862:292–301, 2000.

[17] P. de Groote. A simple calculus of exception handling. In *TLCA 1995*, pages 201–215, 1995.

[18] M. Dezani-Ciancaglini and U. de' Liguoro. Sessions and session types: An overview. In *WS-FM 2009*, pages 1–28, 2009.

[19] J. Epstein, A.P. Black, and S.L. Peyton Jones. Towards haskell in the cloud. In *ACM Haskell Symposium 2011*, pages 118–129, 2011.

[20] A. Fuggetta, G.P. Picco, and G. Vigna. Understanding code mobility. In *IEEE Transactions on Software Engineering*, volume 24, pages 342–361, 1998.

[21] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types.* CUP, 1989.

[22] T.G. Griffin. A formulae-as-type notion of control. In *POPL 1990*, 1990.

[23] Y. Hirai. A lambda calculus for Gödel–dummett logic capturing waitfreedom. In *FLOPS 2012*, pages 151–165, 2012.

[24] W.A. Howard. The formulae-as-types notion of construction. In *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism*, pages 479–491. Academic Press, 1980.

[25] W. Kokke, F. Montesi, and M. Peressotti. Better late than never: A fully-abstract semantics for classical processes. In *POPL*, volume 24, pages 1–29, 2019.

[26] J.-L. Krivine. Lambda-calcul types et modèles. In *Studies in Logic and Foundations of Mathematics*, pages 1–176. Masson, 1990.

[27] P.J. Landin. The mechanical evaluation of expressions. *The Computer Journal*, 6(4):308–320, 1964.

[28] E.G.K. López-Escobar. Implicational logics in natural deduction systems. *The Journal of Symbolic Logic*, 47(1):184–186, 1982.

[29] R. Milner. Lectures on a calculus for communicating systems. In *Seminar on Concurrency*, pages 197–219. Springer, 1984.

[30] R. Milner. Functions as processes. *MSCS*, 2(2):119–141, 1992.

[31] T. Murphy VII, K. Crary, R. Harper, and F. Pfenning. A symmetric modal lambda calculus for distributed computing. In *LICS 2004*, pages 286–295, 2004.

[32] S. Negri. Proof analysis beyond geometric theories: from rule systems to systems of rules. *Journal of Logic and Computation*, 27:513–537, 2016.

[33] M. Parigot. Proofs of strong normalization for second-order classical natural deduction. *The Journal of Symbolic Logic*, 62(4):1461–1479, 1997.

[34] D. Prawitz. Ideas and results in proof theory. In *Proceedings of the Second Scandinavian Logic Symposium*. North-Holland, 1971.

[35] D. Sangiorgi and D. Walker. *The pi-calculus: a Theory of Mobile Processes*. CUP, 2003.

[36] P. Schroeder-Heister. The calculus of higher-level rules, propositional quantification, and the foundational approach to proof-theoretic harmony. *Studia Logica*, 102(6):1185–1216, 2014.

[37] B. Toninho, L. Caires, and F. Pfenning. Higher-order processes, functions, and sessions: A monadic integration. In *ESOP 2013*, pages 350–369, 2013.

[38] P. Wadler. Propositions as sessions. *ICFP 2012*, 24:384–418, 2012.

[39] P. Wadler. Propositions as types. *Communications of the ACM*, 58(12):75–84, 2015.

A Appendix

For the reviewers convenience we write below full proofs of various propositions and lemmas. The idea was to keep in the main text only the non-trivial cases in those proofs.

We present the proof of Proposition 4.2 in full.

A.1 (Parallel Form)**.** *Let $t : A$ be any term. Then $t \mapsto^* t'$, with t' parallel form.*

Proof. By induction on t. As a shortcut, if a term u reduces to a term u' that can be denoted as u'' omitting parentheses, we write $u \rightrightarrows^* u''$.

- $t = x$. Trivial.
- $t = \lambda x\, u$. By the induction hypothesis, $u \rightrightarrows^* u_1 \parallel u_2 \parallel \ldots \parallel u_{n+1}$ and each term u_i, for $1 \le i \le n+1$, is a simply typed λ-term. Applying several permutations we obtain
$$t \rightrightarrows^* \lambda x\, u_1 \parallel \lambda x\, u_2 \parallel \ldots \parallel \lambda x\, u_{n+1}$$
which is the thesis.
- $t = u\, v$. By the induction hypothesis,
$$u \rightrightarrows^* u_1 \parallel u_2 \parallel \ldots \parallel u_{n+1}$$
$$v \rightrightarrows^* v_1 \parallel v_2 \parallel \ldots \parallel v_{m+1}$$
and each term u_i and v_i, for $1 \le i \le n+1, m+1$, is a simply typed λ-term. Applying several permutations we obtain
$$\begin{aligned} t &\rightrightarrows^* (u_1 \parallel u_2 \parallel \ldots \parallel u_{n+1})\, v \\ &\rightrightarrows^* u_1\, v \parallel u_2\, v \parallel \ldots \parallel u_{n+1}\, v \\ &\rightrightarrows^* u_1\, v_1 \parallel u_1\, v_2 \parallel \ldots \parallel u_1\, v_{m+1} \parallel \ldots \\ &\qquad \ldots \parallel u_{n+1}\, v_1 \parallel u_{n+1}\, v_2 \parallel \ldots \parallel u_{n+1}\, v_{m+1} \end{aligned}$$
- $t = \langle u, v \rangle$. By the induction hypothesis,
$$u \rightrightarrows^* u_1 \parallel u_2 \parallel \ldots \parallel u_{n+1}$$
$$v \rightrightarrows^* v_1 \parallel v_2 \parallel \ldots \parallel v_{m+1}$$
and each term u_i and v_i, for $1 \le i \le n+1, m+1$, is a simply typed λ-term. Applying several permutations we obtain
$$\begin{aligned} t &\rightrightarrows^* \langle u_1 \parallel u_2 \parallel \ldots \parallel u_{n+1},\, v \rangle \\ &\rightrightarrows^* \langle u_1, v \rangle \parallel \langle u_2, v \rangle \parallel \ldots \parallel \langle u_{n+1}, v \rangle \\ &\rightrightarrows^* \langle u_1, v_1 \rangle \parallel \langle u_1, v_2 \rangle \parallel \ldots \parallel \langle u_1, v_{m+1} \rangle \parallel \ldots \\ &\qquad \ldots \parallel \langle u_{n+1}, v_1 \rangle \parallel \langle u_{n+1}, v_2 \rangle \parallel \ldots \\ &\qquad \ldots \parallel \langle u_{n+1}, v_{m+1} \rangle \end{aligned}$$
- $t = u\, \pi_i$. By the induction hypothesis,
$$u \rightrightarrows^* u_1 \parallel u_2 \parallel \ldots \parallel u_{n+1}$$
and each term u_i, for $1 \le i \le n+1$, is a simply typed λ-term. Applying several permutations we obtain
$$t \rightrightarrows^* u_1\, \pi_i \parallel u_2\, \pi_i \parallel \ldots \parallel u_{n+1}\, \pi_i.$$

- $t = \iota_i(u)$. By the induction hypothesis,

  $$u \rightrightarrows^* u_1 \parallel u_2 \parallel \dots \parallel u_{n+1}$$

  and each term u_i, for $1 \leq i \leq n+1$, is a simply typed λ-term. Applying several permutations we obtain

  $$t \rightrightarrows^* \iota_i(u_1) \parallel \iota_i(u_2) \parallel \dots \parallel \iota_i(u_{n+1}).$$

- $t = s[x.u, y.v]$. By the induction hypothesis,

  $$\begin{aligned} s &\rightrightarrows^* s_1 \parallel s_2 \parallel \dots \parallel s_{n+1} \\ u &\rightrightarrows^* u_1 \parallel u_2 \parallel \dots \parallel u_{m+1} \\ v &\rightrightarrows^* v_1 \parallel v_2 \parallel \dots \parallel v_{p+1} \end{aligned}$$

  and each term s_i for $1 \leq i \leq n+1$, u_j for $1 \leq j \leq m+1$, and v_k for $1 \leq k \leq p+1$ is a simply typed λ-term. Applying several permutations we obtain

  $$\begin{aligned} t \rightrightarrows^* & s_1[x.u, y.v] \parallel s_2[x.u, y.v] \parallel \dots \parallel s_{n+1}[x.u, y.v] \\ \rightrightarrows^* & s_1[x.u_1, y.v] \parallel s_1[x.u_2, y.v] \parallel \dots \parallel s_1[x.u_{m+1}, y.v] \parallel \\ & s_2[x.u_1, y.v] \parallel s_2[x.u_2, y.v] \parallel \dots \parallel s_2[x.u_{m+1}, y.v] \parallel \\ & \dots \parallel s_{n+1}[x.u_1, y.v] \parallel s_{n+1}[x.u_2, y.v] \parallel \dots \parallel s_{n+1}[x.u_{m+1}, y.v] \\ \rightrightarrows^* & s_1[x.u_1, y.v_1] \parallel s_1[x.u_1, y.v_2] \parallel \dots \parallel s_1[x.u_1, y.v_{p+1}] \parallel \\ & s_1[x.u_2, y.v_1] \parallel s_1[x.u_2, y.v_2] \parallel \dots \parallel s_1[x.u_2, y.v_{p+1}] \parallel \\ & \dots \parallel s_1[x.u_{m+1}, y.v_1] \parallel s_1[x.u_{m+1}, y.v_2] \parallel \dots \parallel s_1[x.u_{m+1}, y.v_{p+1}] \\ & s_2[x.u_1, y.v_1] \parallel s_2[x.u_1, y.v_2] \parallel \dots \parallel s_2[x.u_1, y.v_{p+1}] \parallel \\ & s_2[x.u_2, y.v_1] \parallel s_2[x.u_2, y.v_2] \parallel \dots \parallel s_2[x.u_2, y.v_{p+1}] \parallel \\ & \dots \parallel s_2[x.u_{m+1}, y.v_1] \parallel s_2[x.u_{m+1}, y.v_2] \parallel \dots \parallel s_2[x.u_{m+1}, y.v_{p+1}] \\ & s_{n+1}[x.u_1, y.v_1] \parallel s_{n+1}[x.u_1, y.v_2] \parallel \dots \parallel s_{n+1}[x.u_1, y.v_{p+1}] \parallel \\ & s_{n+1}[x.u_2, y.v_1] \parallel s_{n+1}[x.u_2, y.v_2] \parallel \dots \parallel s_{n+1}[x.u_2, y.v_{p+1}] \parallel \\ & \dots \parallel s_{n+1}[x.u_{m+1}, y.v_1] \parallel s_{n+1}[x.u_{m+1}, y.v_2] \parallel \dots \parallel s_{n+1}[x.u_{m+1}, y.v_{p+1}]. \end{aligned}$$

- $t = u\,\mathsf{efq}_P$. By the induction hypothesis,

  $$u \rightrightarrows^* u_1 \parallel u_2 \parallel \dots \parallel u_{n+1}$$

  and each term u_i, for $1 \leq i \leq n+1$, is a simply typed λ-term. Applying several permutations we obtain

  $$t \rightrightarrows^* u_1\,\mathsf{efq}_P \parallel u_2\,\mathsf{efq}_P \parallel \dots \parallel u_{n+1}\,\mathsf{efq}_P$$

□

We present the proof of Lemma 4.6 in full.

A.2 (Replace!)**.** *Let u be a term in parallel form, v, s be any simply typed λ-terms, τ be the value complexity of v and τ' be the maximum among the complexities of the channel occurrences in v. Then every redex in $u\{v/s\}$ it is either (i) already in v, (ii) of the form $r\{v/s\}$ and has complexity smaller than or equal to the complexity of some redex r of u, or (iii) has complexity τ or is a communication redex of complexity at most τ'.*

Proof. We prove the following stronger statement.

$(*)$ Every redex and channel occurrence in $u\{v/s\}$ it is either (i) already in v, (ii) of the form $r\{v/s\}$ or $aw\{v/s\}$ and has complexity smaller than or equal to the complexity of some redex r or channel occurrence aw of u, or (iii) has complexity τ or τ' or is a communication redex of complexity at most τ'.

We reason by induction on the size and by cases on the possible shapes of the term u.

- $(\lambda x\, t)\, \sigma\, \{v/s\}$, where $\sigma = \sigma_1 \dots \sigma_n$ is any case-free stack. By the induction hypothesis, $(*)$ holds for $t\{v/s\}$ and $\sigma_i\{v/s\}$ where $1 \leq i \leq n$. If $(\lambda x\, t)\, \sigma\{v/s\} = (\lambda x\, t\{v/s\})\, (\sigma\{v/s\})$, all the redexes and channel occurrences that we have to check are either in $t\{v/s\}, \sigma\{v/s\}$ or possibly the head redex, thus the thesis holds. If $(\lambda x\, t)\, \sigma\{v/s\} = v\, (\sigma_i\{v/s\}) \dots (\sigma_n\{v/s\})$, then $v\, (\sigma_i\{v/s\})$ could be a new intuitionistic redex, when $v = \lambda y\, w$, $v = \langle w_1, w_2\rangle$, $v = \iota_i(w)$ or $v = w_0[y_1.w_1, y_2.w_2]$. But the complexity of such a redex is equal to τ.

- $\langle t_1, t_2\rangle\, \sigma\, \{v/s\}$, where $\sigma = \sigma_1 \dots \sigma_n$ is any case-free stack. By the induction hypothesis, $(*)$ holds for $t_i\{v/s\}$ and $\sigma_i\{v/s\}$ where $1 \leq i \leq n$. If $\langle t_1, t_2\rangle\, \sigma\{v/s\} = \langle t_1\{v/s\}, t_2\{v/s\}\rangle\, (\sigma\{v/s\})$, all the redexes and channel occurrences that we have to check are in $t_i\{v/s\}, \sigma\{v/s\}$ or, possibly, the head redex. The former are dealt with using the inductive hypothesis. As for the latter, by Lemma 4.5, the value complexity of $t_i\{v/s\}$ for $i \in \{1, 2\}$ must be at most the value complexity of t_i or exactly τ, thus either (ii) or (iii) holds. If $\langle t_1, t_2\rangle\, \sigma\{v/s\} = v\, (\sigma_i\{v/s\}) \dots (\sigma_n\{v/s\})$, then $v\, (\sigma_i\{v/s\})$ could be a new intuitionistic redex, when $v = \lambda y\, w$, $v = \langle w_1, w_2\rangle$, $v = \iota_i(w)$ or $v = w_0[y_1.w_1, y_2.w_2]$. But the complexity of such a redex is equal to τ.

- $\iota_i(t)\, \sigma\, \{v/s\}$, where $\sigma = \sigma_1 \dots \sigma_n$ is any case-free stack. Obviously σ must be empty since it is case-free. By the induction hypothesis, $(*)$ holds for $t'\{v/s\}$. If $\iota_i(t)\{v/s\} = \iota_i(t\{v/s\})$, all the redexes and channel occurrences that we have to check are in $t\{v/s\}$ and thus the thesis holds. If $\iota_i(t)\{v/s\} = v$ then (i) holds.

- $w_0[z_1.w_1, z_2.w_2]\sigma\{v/s\}$, where σ is any case-free stack. By the induction hypothesis, $(*)$ holds for $w_0\{v/s\}$, $w_1\{v/s\}$, $w_2\{v/s\}$ and $\sigma_i\{v/s\}$ for $1 \leq i \leq n$. If

$$\begin{aligned}&w_0[z_1.w_1, z_2.w_2]\sigma\{v/s\} =\\ &w_0\{v/s\}[z_1.w_1\{v/s\}, z_2.w_2\{v/s\}](\rho\{v/s\})\end{aligned}$$

  we first observe that by Lemma 4.5, the value complexity of $w_0\{v/s\}$ is at most that of w_0 or exactly τ, hence the possible injection or case distinction permutation redex

$$w_0\{v/s\}[z_1.w_1\{v/s\}, z_2.w_2\{v/s\}]$$

  satisfies the thesis. Again by Lemma 4.5, the value complexities of $w_1\{v/s\}$ and $w_2\{v/s\}$ are respectively at most that of w_1 and w_2 or exactly τ. Hence the complexity of the possible case distinction permutation redex

$$(w_0[z_1.w_1\{v/s\}, z_2.w_2\{v/s\}])\sigma_1\{v/s\}$$

  is either τ, and we are done, or at most the value complexity of one among w_1, w_2, thus at most the value complexity of the case distinction permutation redex $(w_0[z_1.w_1, z_2.w_2])\sigma_1$ and we are done.

  If $w_0[z_1.w_1, z_2.w_2]\sigma\{v/s\} = v\, (\sigma_i\{v/s\}) \dots (\sigma_n\{v/s\})$, then there could be a new intuitionistic redex, when $v = \lambda y\, q$, $v = \langle q_1, q_2\rangle$, $v = \iota_i(q)$ or $v = q_0[y_1.q_1, y_2.q_2]$. But the complexity of such a redex is τ.

- $x\,\sigma\{v/s\}$, where x is any simply typed variable and $\sigma = \sigma_1 \ldots \sigma_n$ is any case-free stack. By the induction hypothesis, $(*)$ holds for $\sigma_i\{v/s\}$ where $1 \leq i \leq n$. If $x\,\sigma\{v/s\} = x\,(\sigma\{v/s\})$, all its redexes and channel occurrences are in $\sigma\{v/s\}$, thus the thesis holds. If $x\,\sigma\{v/s\} = v\,(\sigma_i\{v/s\}) \ldots (\sigma_n\{v/s\})$, then $v\,(\sigma_i\{v/s\})$ could be an intuitionistic redex, when $v = \lambda y\, w$, $v = \langle w_1, w_2\rangle$, $v = \iota_i(w)$ or $v = w_0[y_1.w_1, y_2.w_2]$. But the complexity of such a redex is equal to τ.

- $a\,t\,\sigma\{v/s\}$, where a is a channel variable, t a term and $\sigma = \sigma_1 \ldots \sigma_n$ is any case-free stack. By induction hypothesis, $(*)$ holds for $t\{v/s\}, \sigma_i\{v/s\}$ where $1 \leq i \leq n$.

  If $a\,t\,\sigma\{v/s\} = v\,(\sigma_i\{v/s\}) \ldots (\sigma_n\{v/s\})$, then $v\,(\sigma_i\{v/s\})$ could be an intuitionistic redex, when $v = \lambda y\, w$, $v = \langle w_1, w_2\rangle$, $v = \iota_i(w)$ or $v = w_0[y_1.w_1, y_2.w_2]$. But the complexity of such a redex is equal to τ.

  If $a\,t\,\sigma\{v/s\} = a\,(t\{v/s\})(\sigma\{v/s\})$, in order to verify the thesis it is enough to check the complexity of the channel occurrence $a\,(t\{v/s\})$. By Lemma 4.5, the value complexity of $t\{v/s\}$ is at most the value complexity of t or exactly τ, thus either (ii) or (iii) holds.

- ${}_a(t_1 \parallel \ldots \parallel t_m)\{v/s\}$. By the induction hypothesis, $(*)$ holds for $t_i\{v/s\}$ where $1 \leq i \leq m$. The only redex in ${}_a(t_1 \parallel \ldots \parallel t_m)\{v/s\}$ and not in some $t_i\{v/s\}$ can be ${}_a(t_1\{v/s\} \parallel \ldots \parallel t_m\{v/s\})$ itself. But the complexity of such redex equals the maximal complexity of the channel occurrences of the form aw occurring in some $t_i\{v/s\}$, hence it is τ, at most τ' or equal to the complexity of ${}_a(t_1 \parallel \ldots \parallel t_m)$.

□

We present the proof of Lemma 4.7 in full.

A.3 (Eliminate the Case!)**.** *Let u be a term in parallel form. Then for any redex r in*

$$u\{w_i[t/x_i]/\iota_i(t)[x_1.w_1, x_2.w_2]\}$$

of complexity θ, either $\iota_i(t)[x_1.w_1, x_2.w_2]$ has complexity greater than θ; or there is a redex in u of complexity θ which belongs to the same group as r or is a case distinction permutation redex.

Proof. Let $v = w_i[t/x_i]$ and $s = \iota_i(t)[x_1.w_1, x_2.w_2]$. We prove a stronger statement:

$(*)$ For any redex r in $u\{v/s\}$ of complexity θ, either $\iota_i(t)[x_1.w_1, x_2.w_2]$ has complexity greater than θ; or there is a redex in u of complexity θ which belongs to the same group as r or is a case distinction permutation redex. Moreover, for any channel occurrence in $u\{v/s\}$ with complexity θ', either $\iota_i(t)[x_1.w_1, x_2.w_2]$ has complexity greater than θ', or there is an occurrence of the same channel with complexity greater than or equal to θ'.

The proof is by induction on the size of u and by cases according to the possible shapes of u.

- $(\lambda x\, t')\,\sigma\,\{v/s\}$, where $\sigma = \sigma_1 \ldots \sigma_n$ is any case-free stack. By the induction hypothesis, $(*)$ holds for $t'\{v/s\}$ and $\sigma_i\{v/s\}$ for $1 \leq i \leq n$. If $(\lambda x\, t')\,\sigma\{v/s\} = (\lambda x\, t'\{v/s\})\,(\sigma\{v/s\})$, all the redexes and channel occurrences that we have to check are in $t'\{v/s\}$, $\sigma\{v/s\}$ or, possibly, the head redex, thus the thesis holds. Since $s \neq (\lambda x\, t')\,\sigma_1 \ldots \sigma_j$, there is no other possible case.

- $\langle t_1, t_2\rangle\,\sigma\,\{v/s\}$, where $\sigma = \sigma_1 \ldots \sigma_n$ is any case-free stack. By the induction hypothesis, $(*)$ holds for $t_1\{v/s\}$, $t_2\{v/s\}$ and $\sigma_i\{v/s\}$ for $1 \leq i \leq n$.

  If $\langle t_1, t_2\rangle\,\sigma\{v/s\} = \langle t_1\{v/s\}, t_2\{v/s\}\rangle\,(\sigma\{v/s\})$ all the redexes and channel occurrences that we have to check are either in $\sigma\{v/s\}$ or, possibly, the head redex. By Lemma 4.5, the value complexity of $w_i[t/x_i]$ is either at most the value complexity of w_i or the value complexity of t. In the first case, the value complexity of $w_i[t/x_i]$ is at most the value complexity of w_i, which is at most the value complexity of $\iota_i(t)[x_1.w_1, x_2.w_2]$. Thus, by Lemma 4.5, the value complexities of $t_1\{v/s\}, t_2\{v/s\}$ are at most the value complexities respectively of t_1, t_2, thus the value complexity of $\langle t_1\{v/s\}, t_2\{v/s\}\rangle$, and hence that of the possible head redex, is at most the value complexity of $\langle t_1, t_2\rangle$ and we are done. In the second case, the value complexity of $\langle t_1\{v/s\}, t_2\{v/s\}\rangle$, and hence that of the head redex, is either at most the value complexity of $\langle t_1, t_2\rangle$, and we are done, or exactly the value complexity of t, which is smaller than the complexity of the injection redex $\iota_i(t)[x_1.w_1, x_2.w_2]$ occurring in u, which is what we wanted to show. The case in which $\langle t_1, t_2\rangle\, \sigma\{v/s\} = v\,(\sigma_i\{v/s\}) \ldots (\sigma_n\{v/s\})$ is impossible due to the form of $s = \iota_i(t)[x_1.w_1, x_2.w_2]$.

- $\iota_i(t')\,\sigma\,\{v/s\}$, where $\sigma = \sigma_1 \ldots \sigma_n$ is any case-free stack. Obviously σ must be empty since it is case-free. By the induction hypothesis, $(*)$ holds for $t'\{v/s\}$. If $\iota_i(t') = \iota_i(t'\{v/s\})$, all the redexes and channel occurrences that we have to check are in $t'\{v/s\}$ and thus the thesis holds. Indeed, the case in which $\iota_i(t') = v$ is impossible due to the form of $s = \iota_i(t)[x_1.w_1, x_2.w_2]$.

- $x\,\sigma\{v/s\}$, where x is any simply typed variable and $\sigma = \sigma_1 \ldots \sigma_n$ is any case-free stack. By the induction hypothesis, $(*)$ holds for $\sigma_i\{v/s\}$ for $1 \leq i \leq n$. If $x\,\sigma\{v/s\} = x\,(\sigma\{v/s\})$, all its redexes and channel occurrences are in $\sigma\{v/s\}$, thus the thesis holds. The case in which $x\,\sigma\{v/s\} = v\,(\sigma_i\{v/s\}) \ldots (\sigma_n\{v/s\})$ is impossible due to the form of $s = \iota_i(t)[x_1.w_1, x_2.w_2]$.

- $v_0[z_1.v_1, z_2.v_2]\sigma\{v/s\}$, where σ is any case-free stack. By the induction hypothesis, $(*)$ holds for $v_0\{v/s\}$, $v_1\{v/s\}$, $v_2\{v/s\}$ and $\sigma_i\{v/s\}$ for $1 \leq i \leq n$. If

$$\begin{aligned}&v_0[z_1.v_1, z_2.v_2]\sigma\{v/s\} =\\ &v_0\{v/s\}[z_1.v_1\{v/s\}, z_2.v_2\{v/s\}](\rho\{v/s\})\end{aligned}$$

By Lemma 4.5 the value complexity of $w_i[t/x_i]$ is either at most the value complexity of w_i or the value complexity of t. In the first case, the value complexity of $w_i[t/x_i]$ is at most the value complexity of w_i which is at most the value complexity of $\iota_i(t)[x_1.w_1, x_2.w_2]$. Thus, by Lemma 4.5 the value complexities of $v_0\{v/s\}$, $v_1\{v/s\}$, $v_2\{v/s\}$ are at most the value complexity respectively of v_0, v_1, v_2. Hence, the complexity of the possible case distinction permutation redex

$$(v_0[z_1.v_1\{v/s\}, z_2.v_2\{v/s\}])\sigma_1\{v/s\}$$

is at most the complexity of $v_0[z_1.v_1, z_2.v_2]\sigma_1$ and we are done. Moreover, the possible injection or case distinction permutation redex

$$v_0\{v/s\}[z_1.v_1\{v/s\}, z_2.v_2\{v/s\}]$$

satisfies the thesis. In the second case, the value complexities of $v_0\{v/s\}$, $v_1\{v/s\}$, $v_2\{v/s\}$ are at most the value complexities of v_0, v_1, v_2 respectively or exactly the value complexity of t. Hence the complexity of the possible case distinction permutation redex $(v_0[z_1.v_1\{v/s\}, z_2.v_2\{v/s\}])\sigma_1\{v/s\}$ is either at most the value complexity of v_1, v_2, and we are done, or exactly the value complexity of t, which by Proposition 4.1 is at most the complexity of the type of t, thus is smaller than the complexity of the injection redex $\iota_i(t)[x_1.w_1, x_2.w_2]$ occurring in u. Moreover, the possible injection or case distinction permutation redex

$$v_0\{v/s\}[z_1.v_1\{v/s\}, z_2.v_2\{v/s\}]$$

has complexity equal to the value complexity of v_0 or the value complexity of t, and we are done again.

If $v_0[z_1.v_1, z_2.v_2]\sigma\{v/s\} = v\,(\sigma_i\{v/s\})\ldots(\sigma_n\{v/s\})$, then there could be a new intuitionistic redex, when $v = \lambda y\, q$, $v = \langle q_1, q_2\rangle$, $v = \iota_i(q)$ or $v = q_0[y_1.q_1, y_2.q_2]$. If the value complexity of $v = w_i[t/x_i]$ is at most the value complexity of w_i, then the complexity of $w_i[t/x_i](\sigma_i\{v/s\})$ is equal to the complexity of the permutation redex $(\iota_i(t)[x_1.w_1, x_2.w_2])\sigma_i$. If the value complexity of $v = w_i[t/x_i]$ is the value complexity of t, by Proposition 4.1 the complexity of $w_i[t/x_i](\sigma_i\{v/s\})$ is at most the complexity of the type of t, thus is smaller than the complexity of the injection redex $\iota_i(t)[x_1.w_1, x_2.w_2]$ occurring in u and we are done.

- $a\,t'\,\sigma\{v/s\}$, where a is a channel variable, t' a term and $\sigma = \sigma_1 \ldots \sigma_n$ is any case-free stack. By induction hypothesis, $(*)$ holds for t' and $\sigma_i\{v/s\}$ for $1 \leq i \leq n$. Since $s \neq a\,t'\,\sigma_1 \ldots \sigma_j$, the case $a\,t'\,\sigma\{v/s\} = v\,(\sigma_i\{v/s\})\ldots(\sigma_n\{v/s\})$ is impossible.

  If $a\,t'\,\sigma\{v/s\} = a\,(t'\{v/s\})(\sigma\{v/s\})$, in order to verify the thesis it is enough to check the complexity of the channel occurrence $a\,(t'\{v/s\})$. By Lemma 4.5, the value complexity of $w_i[t/x_i]$ is either at most the value complexity of w_i or exactly the value complexity of t. In the first case, the value complexity of $w_i[t/x_i]$ is at most the value complexity of w_i which is at most the value complexity of $\iota_i(t)[x_1.w_1, x_2.w_2]$. Thus, by Lemma 4.5, the value complexity of $t'\{v/s\}$ is at most the value complexity of t' and we are done. In the second case, the value complexity of $t'\{v/s\}$ is the value complexity of t, which by Proposition 4.1 is at most the complexity of the type of t, thus smaller than the complexity of the injection redex $\iota_i(t)[x_1.w_1, x_2.w_2]$ occurring in u, which is what we wanted to show.

- ${}_a(t_1 \parallel \ldots \parallel t_m)\{v/s\}$. By the induction hypothesis, $(*)$ holds for $t_i\{v/s\}$ for $1 \leq i \leq m$. The only redex in ${}_a(t_1 \parallel \ldots \parallel t_m)\{v/s\}$ and not in some $t_i\{v/s\}$ can be ${}_a(t_1\{v/s\} \parallel \ldots \parallel t_m\{v/s\})\{v/s\}$ itself. But the complexity of such redex equals the maximal complexity of the occurrences of the channel a in the $t_i\{v/s\}$. Hence the statement follows.

□

We present the proof of Lemma 4.8 in full.

Lemma A.4 (In the Case). *Let u be a term in parallel form. Then for any redex r_1 of Group 1 in $u\{t[x_1.v_1\xi, x_2.v_2\xi]/t[x_1.v_1, x_2.v_2]\xi\}$, where ξ is a stack of length 1, there is a redex in u with complexity greater than or equal to r_1; for any redex r_2 of Group 2 in $u\{t[x_1.v_1\xi, x_2.v_2\xi]/t[x_1.v_1, x_2.v_2]\xi\}$, there is a redex of Group 2 in u with complexity greater than or equal to r_2.*

Proof. Let $v = t[x_1.v_1\xi, x_2.v_2\xi]$ and $s = t[x_1.v_1, x_2.v_2]\xi$. We prove the following stronger statement.

$(*)$ For any redex r_1 of Group 1 in $u\{t[x_1.v_1\xi, x_2.v_2\xi]/t[x_1.v_1, x_2.v_2]\xi\}$ there is a redex in u with complexity greater than or equal to r_1; for any redex r_2 of Group 2 in $u\{t[x_1.v_1\xi, x_2.v_2\xi]/t[x_1.v_1, x_2.v_2]\xi\}$ there is a redex of Group 2 in u with greater or equal complexity than r_2. Moreover, for any channel occurrence in $u\{v/s\}$ with complexity θ', there is in u an occurrence of the same channel with complexity greater or equal than θ'.

We first observe that the possible Group 1 redexes $v_1\xi$ and $v_2\xi$ have at most the complexity of the case distinction permutation $t[x_1.v_1, x_2.v_2]\xi$. The rest of the proof is by induction on the size of u.

- $(\lambda x\, t')\, \sigma\, \{v/s\}$, where $\sigma = \sigma_1 \dots \sigma_n$ is any case-free stack. By the induction hypothesis, $(*)$ holds for $t'\{v/s\}$ and $\sigma_i\{v/s\}$ where $1 \le i \le n$. If $(\lambda x\, t')\, \sigma\{v/s\} = (\lambda x\, t'\{v/s\})\, (\sigma\{v/s\})$, all the redexes and channel occurrences that we have to check are either in $\sigma\{v/s\}$ or, possibly, the head redex, thus the thesis holds. If

$$(\lambda x\, t')\, \sigma\{v/s\} = t[x_1.v_1\xi, x_2.v_2\xi]\, (\sigma_i\{v/s\}) \dots (\sigma_n\{v/s\})$$

  then $t[x_1.v_1\xi, x_2.v_2\xi](\sigma_i\{v/s\})$ can be a new permutation redex. If ξ is case free, this redex has complexity 0 by Lemma 4.4 and we are done. If ξ is not case free, $t[x_1.v_1\xi, x_2.v_2\xi](\sigma_i\{v/s\})$ has the same complexity as the permutation $(t[x_1.v_1, x_2.v_2]\xi)\sigma_i$.

- $\langle t_1, t_2\rangle\, \sigma\, \{v/s\}$, where $\sigma = \sigma_1 \dots \sigma_n$ is any case-free stack. By the induction hypothesis, $(*)$ holds for $t_1\{v/s\}$, $t_2\{v/s\}$ and $\sigma_i\{v/s\}$ where $1 \le i \le n$. If

$$\langle t_1, t_2\rangle\, \sigma\{v/s\} = \langle t_1\{v/s\}, t_2\{v/s\}\rangle\, (\sigma\{v/s\})$$

  all the redexes and channel occurrences that we have to check are either in $\sigma\{v/s\}$ or, possibly, the head redex. The former are dealt with using the inductive hypothesis. As for the latter, it is immediate to see that the value complexity of $t[x_1.v_1\xi, x_2.v_2\xi]$ is equal to the value complexity of $t[x_1.v_1, x_2.v_2]\xi$. By Lemma 4.5, the value complexity of $t_i\{v/s\}$ is at most that of t_i, and we are done. If $\langle t_1, t_2\rangle\, \sigma\{v/s\} = v\, (\sigma_i\{v/s\}) \dots (\sigma_n\{v/s\})$, then $v\, (\sigma_i\{v/s\}) = t[x_1.v_1\xi, x_2.v_2\xi](\sigma_i\{v/s\})$ can be a new permutation redex. If ξ is case free, this redex has complexity 0 by Lemma 4.4 and we are done. Otherwise,

$$t[x_1.v_1\xi, x_2.v_2\xi](\sigma_i\{v/s\})$$

  has the same complexity as $(t[x_1.v_1, x_2.v_2]\xi)\sigma_i$.

- $\iota_i(t')\, \sigma\, \{v/s\}$, where $\sigma = \sigma_1 \dots \sigma_n$ is any case-free stack. Obviously, σ must be empty since it is case-free. By the induction hypothesis, $(*)$ holds for $t'\{v/s\}$. If $\iota_i(t') = \iota_i(t'\{v/s\})$, all the redexes and channel occurrences that we have to check are in $\sigma\{v/s\}$ and thus the thesis holds.

- $w_0[z_1.w_1, z_2.w_2]\sigma\{v/s\}$, where σ is any case-free stack. By the induction hypothesis, $(*)$ holds for $w_0\{v/s\}$, $v_1\{v/s\}$, $v_2\{v/s\}$ and $\sigma_i\{v/s\}$ for $1 \le i \le n$. If

$$\begin{aligned}&w_0[z_1.w_1, z_2.w_2]\sigma\{v/s\} = \\ &w_0\{v/s\}[z_1.w_1\{v/s\}, z_2.w_2\{v/s\}](\sigma\{v/s\})\end{aligned}$$

  Since the value complexity of $t[x_1.v_1\xi, x_2.v_2\xi]$ is equal to the value complexity of $t[x_1.v_1, x_2.v_2]\xi$, by Lemma 4.5 the value complexities of $w_0\{v/s\}$, $w_1\{v/s\}$ and $w_2\{v/s\}$ are respectively at most that of w_0, w_1 and w_2. The complexity of the possible case distinction permutation redex $(w_0\{v/s\}[z_1.w_1\{v/s\}, z_2.w_2\{v/s\}])\sigma_1\{v/s\}$ is thus at most the value complexity of w_1, w_2 respectively, and hence the complexity of the case permutation redex $(w_0[z_1.w_1, z_2.w_2])\sigma_1$. Moreover, the possible injection or case permutation redex

$$w_0\{v/s\}[z_1.w_1\{v/s\}, z_2.w_2\{v/s\}]$$

  has complexity equal to the value complexity of w_0 and we are done.

If $w_0[z_1.w_1, z_2.w_2]\sigma\{v/s\} = v\,(\sigma_i\{v/s\})\dots(\sigma_n\{v/s\})$, then there could be a new case distinction permutation redex because $v = t[x_1.v_1\xi, x_2.v_2\xi]$. If ξ is case free, by Lemma 4.4, this redex has complexity 0 and we are done; if not, it has the same complexity as $t[x_1.v_1, x_2.v_2]\xi\sigma_1$ and we are done again.

- $x\,\sigma\{v/s\}$, where x is any simply typed variable and $\sigma = \sigma_1 \dots \sigma_n$ is any case-free stack. By induction hypothesis, $(*)$ holds for $\sigma_i\{v/s\}$ where $1 \le i \le n$. If $x\,\sigma\{v/s\} = x\,(\sigma\{v/s\})$, all its redexes and channel occurrences are in $\sigma\{v/s\}$, thus the thesis holds. If

$$x\,\sigma\{v/s\} = t[x_1.v_1\xi, x_2.v_2\xi]\,(\sigma_i\{v/s\})\dots(\sigma_n\{v/s\})$$

  then $v\,(\sigma_i\{v/s\})$ can be a new permutation redex. If ξ is case free, this redex has complexity 0 and we are done. Otherwise, $t[x_1.v_1\xi, x_2.v_2\xi](\sigma_i\{v/s\})$ has the same complexity as $t[x_1.v_1, x_2.v_2]\xi\sigma_i$.

- $a\,t\,\sigma\{v/s\}$, where a is a channel variable, t a term and $\sigma = \sigma_1 \dots \sigma_n$ is any case-free stack. By induction hypothesis, $(*)$ holds for t and $\sigma_i\{v/s\}$ where $1 \le i \le n$.

  If $a\,t\,\sigma\{v/s\} = a\,(t\{v/s\})(\sigma\{v/s\})$, in order to verify the thesis it is enough to check the complexity of the channel occurrence $a\,(t\{v/s\})$. Since the value complexity of $t[x_1.v_1\xi, x_2.v_2\xi]$ is equal to the value complexity of $t[x_1.v_1, x_2.v_2]\xi$, by Lemma 4.5 the value complexity of $t\{v/s\}$ is at most that of t, and we are done.

  If $a\,t\,\sigma\{v/s\} = a\,t\,[x_1.v_1\xi, x_2.v_2\xi]\,(\sigma_i\{v/s\})\dots(\sigma_n\{v/s\})$ then $v\,(\sigma_i\{v/s\})$ can be a new permutation redex. If ξ is case free, this redex has complexity 0 and we are done. Otherwise, $t[x_1.v_1\xi, x_2.v_2\xi](\sigma_i\{v/s\})$ has the same complexity as $(t[x_1.v_1, x_2.v_2]\xi)\sigma_i$.

- ${}_a(t_1 \parallel \dots \parallel t_m)\{v/s\}$. By the induction hypothesis, $(*)$ holds for $t_i\{v/s\}$ where $1 \le i \le m$. The only redex in ${}_a(t_1 \parallel \dots \parallel t_m)\{v/s\}$ and not in some $t_i\{v/s\}$ can be ${}_a(t_1\{v/s\} \parallel \dots \parallel t_m\{v/s\})$ itself. But the complexity of such redex equals the maximal complexity of the occurrences of the channel a in $t_i\{v/s\}$. Hence the statement follows.

□

We present the proof of Proposition 5.1.

Proposition A.5 (Parallel Form Property)**.** *If $t \in \mathrm{NF}$ is a λ_{L}-term, then it is in parallel form.*

Proof. By induction on the structure of t. The case t is a variable is trivial.

- $t = \lambda x\, v$. Since t is normal, v cannot be of the form ${}_a(u_1 \parallel \dots \parallel u_m)$, otherwise one could apply the permutation

$$t = \lambda x^A\, {}_a(u_1 \parallel \dots \parallel u_m) \mapsto {}_a(\lambda x^A\, u_1 \parallel \dots \parallel \lambda x^A.u_m)$$

  and t would not be in normal form. Hence, by i.h. v must be a simply typed λ-term.

- $t = \langle v_1, v_2\rangle$. Since t is normal, neither v_1 nor v_2 can be of the form ${}_a(u_1 \parallel \dots \parallel u_m)$, otherwise one could apply one of the permutations

$$\langle {}_a(u_1 \parallel \dots \parallel u_m),\, w\rangle \mapsto {}_a(\langle u_1, w\rangle \parallel \dots \parallel \langle u_m, w\rangle)$$

$$\langle w,\, {}_a(u_1 \parallel \dots \parallel u_m)\rangle \mapsto {}_a(\langle w, u_1\rangle \parallel \dots \parallel \langle w, u_m\rangle)$$

  and t would not be in normal form. Hence, by i.h. v_1 and v_2 must be simply typed λ-terms.

- $t = v_1\, v_2$. Since t is normal, neither v_1 nor v_2 can be of the form ${}_a(u_1 \parallel \ldots \parallel u_m)$, otherwise one could apply one of the permutations

$$ {}_a(u_1 \parallel \ldots \parallel u_m)\, w \mapsto {}_a(u_1 w \parallel \ldots \parallel u_m w) $$

$$ w\, {}_a(u_1 \parallel \ldots \parallel u_m) \mapsto {}_a(w u_1 \parallel \ldots \parallel w u_m) $$

  and t would not be in normal form. Hence, by i.h. v_1 and v_2 must be simply typed λ-terms.

- $t = v\, \mathsf{efq}_P$. Since t is normal, v cannot be of the form ${}_a(u_1 \parallel \ldots \parallel u_m)$, otherwise one could apply the permutation

$$ {}_a(u_1 \parallel \ldots \parallel u_m)\, \mathsf{efq}_P \mapsto {}_a(u_1\, \mathsf{efq}_P \parallel \ldots \parallel u_m\, \mathsf{efq}_P) $$

  and t would not be in normal form. Hence, by the induction hypothesis $u_1, \ldots, u_m$ must be simply typed λ-terms.

- $t = u\, \pi_i$. Since t is normal, v can be of the form ${}_a(u_1 \parallel \ldots \parallel u_m)$, otherwise one could apply the permutation

$$ {}_a(u_1 \parallel \ldots \parallel u_m)\, \pi_i \mapsto {}_a(u_1 \pi_i \parallel \ldots \parallel u_m \pi_i) $$

  and t would not be in normal form. Hence, by induction hypothesis u must be a simply typed λ-term, which is the thesis.

- $t = {}_a(u_1 \parallel \ldots \parallel u_m)$. By the induction hypothesis the thesis holds for u_i where $1 \leq i \leq m$ and hence trivially for t.

□