ELSEVIER

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Theoretical Computer Science 323 (2004) 71–127

Theoretical Computer Science

www.elsevier.com/locate/tcs

# Pattern matching as cut elimination

Serenella Cerrito[a], Delia Kesner[b,*]

[a] *LaMI (CNRS UMR 8042) Université d'Evry, Val d'Essonne, 523, place des terrasses de l'agora, 91000 Evry Cedex, France*

[b] *PPS (CNRS UMR 7126), Université Paris VII, France*

Received 2 July 2001; received in revised form 27 February 2003; accepted 9 March 2004
Communicated by J. Tiuryn

**Abstract**

We present a typed pattern calculus with *explicit pattern matching* and *explicit substitutions*, where both the typing rules and the reduction rules are modeled on the same logical proof system, namely *Gentzen sequent calculus* for intuitionistic minimal logic. Our calculus is inspired by the Curry–Howard Isomorphism, in the sense that types, both for patterns and terms, correspond to propositions, terms correspond to proofs, and term reduction corresponds to sequences of sequent proof normalization steps performed by cut elimination. The calculus enjoys subject reduction, confluence, preservation of strong normalization w.r.t a system with meta-level substitutions and strong normalization for well-typed terms. As a consequence, it can be seen as an implementation calculus for functional formalisms defined with meta-level operations for pattern matching and substitutions. This work is a revised and extended version of Cerrito and Kesner (14th Annual IEEE Symposium on Logic in Computer Science (LICS), IEEE Computer Society Press, Silver Spring, MD, 1999, pp. 98–108).


## 1. Introduction

In this paper we propose a Typed Pattern Calculus with Explicit Substitutions, called $TPC_{ES}$, where both the typing rules and the reduction rules are modeled on the same logical proof system, namely sequent calculus for intuitionistic minimal logic. The formalism $TPC_{ES}$ is inspired by the Curry–Howard Isomorphism, which relates a logical

* Corresponding author. Tel.: +33-1-4427-9961; fax: +33-1-4427-8654.
*E-mail addresses:* serena@lami.univ-evry.fr (S. Cerrito), kesner@pps.jussieu.fr (D. Kesner).

0304-3975/$ - see front matter © 2004 Published by Elsevier B.V.
doi:10.1016/j.tcs.2004.03.032

proof system with a typed programming language, and can be roughly described by the slogan:

> *Proofs are Terms (Programs), Formulae are Types, Proof-Normalization is Program-Evaluation.*

The Curry–Howard isomorphism is a powerful tool which has been proven very fruitful with respect to both Computer Science and Logic. In Computer Science it has been proven useful, for instance, in synthesis of programs and design of languages. In Logic, it has provided insights on the understanding of constructive logic proof systems.

On the other hand, more and more works have been developed these last years in higher-order calculi taking into account the *explicit substitutions* formalisms [1,6,7,27] and the, modelization of *explicit pattern-matching* constructs [11]. Pattern matching is a mechanism provided by all modern functional languages (Hope [8], SML [33], Miranda [35], Caml [9] and Haskell [24]), and most of current proof assistants (Coq [12], PVS [34], HOL [23], LEGO [31] and ALF [3]) since it gives a very natural way to define functions (resp. proofs) by cases.

By including explicit pattern matching as well as explicit substitutions in a calculus, one gets a fine control of these key constructors which are usually used to implement languages, thereby obtaining a flexible tool to reason about evaluation *strategies* for both pattern-matching and substitution.

Since logical proof system underlying typing and reduction rules of $TPC_{\mathrm{ES}}$ is *Gentzen sequent calculus*, rather than natural deduction, thus proof normalization is *cut elimination*. As a consequence, pattern matching and substitution computations of $TPC_{\mathrm{ES}}$, being embedded in the calculus via an *explicit* encoding (rather than being meta-level operations), correspond to sequences of cut elimination steps. We use sequent right rules to build and type *terms*, left rules to build and type nested *patterns* and the *cut rule* to model a general *let* construct (which is the starting point of computations), as well as an explicit substitution constructor. The structural rules left contraction and left weakening are used to model the layered and wildcard patterns of functional languages.

In contrast to [28], which presents a calculus inspired from Gentzen sequent calculus that models programs with meta-level pattern-matching and substitution, we keep both operations as internal (or explicit). Also, the notion of reduction in [28] does not correspond to normalization of sequent proofs, but to normalization of proofs in natural deduction. The novelty of our approach is the design of a typed pattern calculus based on a computational interpretation of the Gentzen sequent proofs, thereby allowing to model pattern matching via cut elimination.

Pattern matching is modeled in $TPC_{\mathrm{ES}}$ by reduction of terms having the form $\mathtt{let}\ M\ \mathtt{be}\ P:A\ \mathtt{in}\ N$ where $M$ and $N$ are terms and $P$ is a *complex* pattern, while explicit substitution computation is modeled by reduction of terms having the form $N[x/M]$, where $x$ is a *variable* pattern. In the first case, a reduction step "decomposes" the pattern $P:A$ and the corresponding term $M$ and corresponds to a cut elimination step replacing a cut over a non-atomic formula by cuts on its sub-formulae; for instance, matching of $\langle M_1, M_2\rangle$ with $\langle P_1, P_2\rangle : A_1 \times A_2$ reduces to matching $M_1$ with $P_1 : A_1$ and matching $M_2$ with $P_2 : A_2$ and corresponds to the cut elimination step replacing a cut

over the formula $A_1 \wedge A_2$ by cuts on $A_1$ and $A_2$. The reader accustomed with cut elimination procedures will recognize that this corresponds exactly to a "principal–principal" cut case, called also "key case". In the second case, the reduction step executes the substitution of the variable in the term $N$, either by distributing it over $N$ or else by physically replacing the variable $x$ by the term $M$ when $N$ is a variable. Also in this case the reduction corresponds to a step in the procedure of eliminating cuts in sequent proofs (a "non-key-case"). For example, let us consider the following cut elimination step:

$$\dfrac{\dfrac{\mathscr{D}}{\Gamma \vdash A} \quad \Gamma, A \vdash A\ (axiom)}{\Gamma \vdash A}\ (cut) \Longrightarrow \dfrac{\mathscr{D}}{\Gamma \vdash A}.$$

Now, let us interpret proofs by terms and propositions by types as suggested by the Curry-Howard correspondence. We then get the following transformation of typing derivations:

$$\dfrac{\Gamma \vdash M : A \quad \Gamma, x : A \vdash x : A\ (axiom)}{\Gamma \vdash x[x/M] : A}\ (sub) \quad \Longrightarrow \quad \Gamma \vdash M : A$$

which suggests that the process of cut elimination consists in reducing the term $x[x/M]$ to the term $M$, exactly as in the $Var1$ reduction rule in [1,6,7,27] defined by: $x[x/M] \longrightarrow M$.

As far as we know, our proposal turns out to be the first rational reconstruction of pattern matching in terms of *cut elimination*.

We prove that the $TPC_{\mathrm{ES}}$ calculus enjoys the properties of subject reduction, confluence, and strong normalization for *well-typed* terms. Moreover, the calculus enjoys preservation of strong normalization w.r.t. another system *TPC*, which is a property on *all pure* terms (i.e. terms not having explicit substitutions). The study of this property in the framework of calculi with explicit substitutions has received much attention since Melliès [32] has shown that $\lambda_\sigma$-calculus [1] does not preserve strong normalization of $\lambda$-calculus. Under a quite natural definition of *value*, we show that it is possible to evaluate any given (functional) *program* (modeled by a $TPC_{\mathrm{ES}}$ *closed* term), to a *result* (modeled by a value).

This paper is a revised and extended version of [10]. It is organized as follows. In Section 2 we introduce the syntax, the typing rules and the reduction rules of $TPC_{\mathrm{ES}}$. We also prove the subject reduction property for $TPC_{\mathrm{ES}}$ in Section 3. Section 4 shows how to code simply typed $\lambda$ calculus into $TPC_{\mathrm{ES}}$ and how to extend $TPC_{\mathrm{ES}}$ to recursive types. In Section 5 we define another pattern calculus, *TPC*; proving some properties of *TPC* helps us to prove certain properties of $TPC_{\mathrm{ES}}$, namely confluence and strong normalization, which are established in Section 6. In this last section we also establish the reducibility of closed typed $TPC_{\mathrm{ES}}$ terms to values. Finally, in Section 7, we conclude and we compare our approach with other works.

Throughout the paper we will use standard notations from rewriting that we borrow from [15].

## 2. The calculus $TPC_{\mathrm{ES}}$

### 2.1. Raw syntax

The set of *types* is defined by the following grammar, where $\iota$ ranges over base types:

$$A ::= \iota \mid A \times A \text{ (product)} \mid A + A \text{ (sum)} \mid A \to A \text{ (functional)}.$$

We distinguish two disjoint sets of variables: *usual variables* noted $x, y, v, z, \ldots$ and *sum variables* noted $\xi, \rho, \psi, \ldots$, etc. Sum variables play an essential role in patterns of sum type and make it possible to define functions by cases.

*(Raw) Patterns* ($P$) are defined by the following grammar:

| | | |
|---|---|---|
| $P ::=$ | $\_$ | wildcard |
| | $\mid x$ | variable |
| | $\mid \sharp x$ | arrow pattern |
| | $\mid \langle P, P \rangle$ | product pattern |
| | $\mid (P \mid_\xi P)$ | sum pattern |
| | $\mid @(P, P)$ | contraction pattern |

*(Raw) Terms* ($M$) are defined by the following grammar:

| | | |
|---|---|---|
| $M ::=$ | $x$ | variable |
| | $\mid \langle M, M \rangle$ | pair |
| | $\mid \mathtt{inl}_A(M)$ | left injection |
| | $\mid \mathtt{inr}_A(M)$ | right injection |
| | $\mid M[x/M]$ | explicit substitution |
| | $\mid \lambda P{:}A.M$ | lambda abstraction |
| | $\mid [M \mid_\xi M]$ | case |
| | $\mid \mathtt{let}\ M\ \mathtt{be}\ P{:}A\ \mathtt{in}\ M$ | let |
| | $\mid z\ \mathtt{of}\ M\ \mathtt{is}\ P{:}A\ \mathtt{in}\ M$ | variable application |
| | $\mid \lambda P{:}A.M\ \mathtt{of}\ M\ \mathtt{is}\ P{:}A\ \mathtt{in}\ M$ | lambda application |

where $\xi$ is a sum variable, $x$ and $z$ are usual variables.

A (usual) variable $z$ in an application term of the form $z\ \mathtt{of}\ M\ \mathtt{is}\ P{:}A\ \mathtt{in}\ M$ is said to be an *application variable*. Both sum variables and application variables are also called *communication variables*; the intuitive reason of such a terminology will be explained in Section 2.2.

Note that an *application variable* $z$ is not just a variable of functional type, but a functional variable appearing in a particular context $z\ \mathtt{of}\ M\ \mathtt{is}\ P{:}A\ \mathtt{in}\ M$.

We use the notation $\mathscr{T}_{TPC_{\mathrm{ES}}}$ to denote the set of *raw* terms defined above.

A term is called *pure* if none of its sub-terms is of the form $M[\_/\_]$, that is, if it has no explicit substitutions.

Free and bound occurrences of variables of terms are defined as usual, with the understanding that the terms of the form $\lambda P : A.M$, $U$ `of` $N$ `is` $P : A$ `in` $M$, `let` $N$ `be` $P : A$ `in` $M$ (resp. $M[x/N]$) define bindings whose scope is $M$ for all the variables occurring in $P$ (resp. for $x$). We denote by $Var(P)$ the set of variables occurring in the pattern $P$ and by $FV(M)$ the set of free variables occurring in the term $M$. They can be defined by induction as follows:

$$
\begin{aligned}
Var(\_) &= \emptyset,\\
Var(x) &= \{x\},\\
Var(\sharp z) &= \{z\},\\
Var(\langle P, Q\rangle) &= Var(P) \cup Var(Q),\\
Var((P \mid_\xi Q)) &= Var(P) \cup Var(Q) \cup \{\xi\},\\
Var(@(P, Q)) &= Var(P) \cup Var(Q),
\end{aligned}
$$

$$
\begin{aligned}
FV(x) &= \{x\},\\
FV(\langle M, N\rangle) &= FV(M) \cup FV(N),\\
FV(\mathtt{inl}_A(M)) &= FV(M),\\
FV(\mathtt{inr}_A(M)) &= FV(M),\\
FV(M[x/N]) &= FV(N) \cup (FV(M) \setminus \{x\}),\\
FV(\lambda P{:}A.M) &= FV(M) \setminus Var(P),\\
FV([M \mid_\xi N]) &= FV(M) \cup FV(N) \cup \{\xi\},\\
FV(\mathtt{let}\ N\ \mathtt{be}\ P{:}A\ \mathtt{in}\ M) &= FV(N) \cup (FV(M) \setminus Var(P)),\\
FV(U\ \mathtt{of}\ N\ \mathtt{is}\ Q{:}A\ \mathtt{in}\ M) &= FV(U) \cup FV(N) \cup (FV(M) \setminus Var(Q)).
\end{aligned}
$$

We work modulo $\alpha$-conversion so that renaming of (usual or communication) bound variables is always used to avoid clash of variables. The meta-expression $M\{x \leftarrow N\}$ denotes the term $M$ where all the free occurrences of $x$ have been replaced by the term $N$.

The set of *free communication variables* of $M$, written $FCV(M)$, is the subset of $FV(M)$ that only contains communication variables. Thus, for example if $N = \lambda(x_1 \mid_\xi x_2) : A + A \to A.[x_1 \mid_\xi x_2]$, then the set of free communication variables of the term $z$ `of` $N$ `is` $w{:}B$ `in` ($w$ `of` $[z_1 \mid_\zeta z_2]$ `is` $y{:}C$ `in` $y$) is $\{z, \zeta\}$.

A pattern $P$ is said to be *linear* if every variable occurs at most once in $P$. The set of *acceptable patterns* of type $A$, denoted by $AP(A)$, is defined as the smallest set of linear patterns verifying the following properties: $\_ \in AP(A)$; $x \in AP(A)$ for any variable $x$; $\sharp x \in AP(B \to C)$ for any variable $x$; $@(P, Q) \in AP(A)$ if $P \in AP(A)$ and $Q \in AP(A)$; $\langle P, Q\rangle \in AP(B \times C)$ and $(P \mid_\xi Q) \in AP(B + C)$ if $P \in AP(B)$ and $Q \in AP(C)$. As an example, $\langle y, \sharp x\rangle \in AP(B \times (C \to D))$ but $\sharp x \notin AP(B \times C)$. The role of the notion of "acceptable patterns" is to prevent the typing rules to introduce meaningless pattern expressions such as for example a pattern $\langle x, y\rangle$ of type $A + B$.

*Typing judgments* have the form $\Gamma \triangleright M : A$ where $\Gamma$, called a *pattern type assignment*, is a multi-set of elements of the form $P : A$ (where $P$ is a pattern and $A$ is a type), so the order of the patterns in a pattern type assignment is not relevant. The order of patterns in $@(P_1, P_2)$ is also irrelevant. We write $Var(\Gamma)$ to denote the set $\bigcup_{P:A \in \Gamma} Var(P)$. As for patterns, a pattern type assignment $\Gamma$ is said to be *linear* if every variable occurs at most once in $\Gamma$. Note that even linear pattern type assignments

may be multi-sets when they contain several occurrences of a wildcard pattern as for example $x:A$, $\_:B$, $\_:B$.

Below, we define the notions of replacement of a sum communication variable by a constant $\mathtt{L}$ or $\mathtt{R}$ for a pattern type assignment.

**Definition 2.1** (Replacement in a Pattern). The replacement of a sum communication variable by $\mathtt{K}$ associates to a pattern $P:A$ a new pattern $P:A_{(\xi,\mathtt{K})}$, where $\mathtt{K}\in\{\mathtt{L},\mathtt{R}\}$. It is defined as follows:

$$
\begin{array}{lll}
x:A_{(\xi,\mathtt{K})} & = x:A, & \\
\_:A_{(\xi,\mathtt{K})} & = \_:A, & \\
\sharp z:A\to B_{(\xi,\mathtt{K})} & = \sharp z:A\to B, & \\
\langle P,Q\rangle:A\times B_{(\xi,\mathtt{K})} & = \langle P_{(\xi,\mathtt{K})},Q_{(\xi,\mathtt{K})}\rangle:A\times B, & \\
(P\mid_{\xi} Q):A+B_{(\xi,\mathtt{K})} & = P_{(\xi,\mathtt{L})}:A & \text{if } \mathtt{K} \text{ is } \mathtt{L},\\
(P\mid_{\xi} Q):A+B_{(\xi,\mathtt{K})} & = Q_{(\xi,\mathtt{R})}:B & \text{if } \mathtt{K} \text{ is } \mathtt{R},\\
(P\mid_{\delta} Q):A+B_{(\xi,\mathtt{K})} & = (P_{(\xi,\mathtt{K})}\mid_{\delta} Q_{(\xi,\mathtt{K})}):A+B & \text{if } \xi\neq\delta,\\
@(P,Q):A_{(\xi,\mathtt{K})} & = @(P_{(\xi,\mathtt{K})},Q_{(\xi,\mathtt{K})}):A. &
\end{array}
$$

It is easy to remark that $Var(P_{(\xi,\mathtt{K})})\subseteq Var(P)$ and that $P_{(\xi,\mathtt{K})}=P$ whenever $\xi\notin Var(P)$.

This notion extends naturally to a pattern type assignment $\Gamma=P_1:A_1,\ldots,P_n:A_n$ by defining $\Gamma_{(\xi,\mathtt{K})}$ as the pattern type assignment $P_1:A_{1(\xi,\mathtt{K})},\ldots,P_n:A_{n(\xi,\mathtt{K})}$.

### 2.2. Typing rules

The syntax for terms and patterns that we have introduced in Section 2.1 is referred to as *raw*, to emphasize the fact that it may or may not type-check. For example, raw patterns are not necessarily linear but the well-typed ones are.

The typing rules are presented in Fig. 1. All the pattern type assignments that can be built using the typing rules are linear. In the $(proj)$ rule the $x_j$'s are all distinct. In the $(+left)$ rule $\xi$ is a fresh sum variable and $(P\mid_{\xi} Q)$ is a linear pattern. In the $(\to left)$ rule $z$ is a fresh variable. In the $(\to left)$, $(app)$, $(let)$ and $(sub)$ rules we require the condition $FCV(M)=\emptyset$. Also, in the $(subf)$ rule we require $FCV(\lambda P:A.M)=\emptyset$. This is essential to guarantee the subject reduction property. In $(wildcard)$, we require $P$ to be in $AP(A)$ and $P:A,\Gamma$ to be linear. In the $(app)$ rule we also ask $FCV(\lambda P:A.J)=\emptyset$.

In the sequel, sometimes we will make reference to both the *sub* and *subf* rule as instances of the following generic substitution rule $sub*$:

$$\frac{\Gamma\triangleright M:A \qquad ?v:A,\Gamma\triangleright N:B}{\Gamma\triangleright N[v/M]:B}\quad (sub*)$$

where the meta-notation $?v$ is used to denote either a variable pattern $v$ or an arrow pattern $\sharp v$, and the typing rule is subject to the following restrictions:

1. $FCV(M)=\emptyset$
2. If $?v$ is an arrow pattern $\sharp v$, then $M$ has necessarily the form $\lambda P:C.L$.

$$x_1:A_1,\dots x_n:A_n \rhd x_i:A_i \quad (proj)$$

$$\frac{P:A,\Gamma\rhd M:C \qquad Q:B,\Gamma\rhd N:C}{(P\mid_\xi Q):A+B,\Gamma\rhd[M\mid_\xi N]:C}\ (+left)$$

$$\frac{\Gamma\rhd\lambda P:AJ:A\to B \qquad \Gamma\rhd M:A \qquad \Gamma,Q:B\rhd N:C}{\Gamma\rhd\lambda P:AJ\ \texttt{of}\ M\ \texttt{is}\ Q:B\ \texttt{in}\ N:C}\ (app)$$

$$\frac{P:A,Q:B,\Gamma\rhd M:C}{\langle P,Q\rangle:A\times B,\Gamma\rhd M:C}\ (\times left) \qquad \frac{\Gamma\rhd M:A \qquad \Gamma\rhd N:B}{\Gamma\rhd\langle M,N\rangle:A\times B}\ (\times right)$$

$$\frac{\Gamma\rhd M:A}{\Gamma\rhd \texttt{inl}_B(M):A+B}\ (+right1) \qquad \frac{\Gamma\rhd N:B}{\Gamma\rhd \texttt{inr}_A(N):A+B}\ (+right2)$$

$$\frac{P:A,\Gamma\rhd M:B}{\Gamma\rhd\lambda P:A.M:A\to B}\ (\to right) \qquad \frac{\Gamma\rhd M:A \qquad Q:B,\Gamma\rhd N:C}{\sharp z:A\to B,\Gamma\rhd z\ \texttt{of}\ M\ \texttt{is}\ Q:B\ \texttt{in}\ N:C}\ (\to left)$$

$$\frac{\Gamma\rhd M:A \qquad P:A,\Gamma\rhd N:B}{\Gamma\rhd \texttt{let}\ M\ \texttt{be}\ P:A\ \texttt{in}\ N:B}\ (let) \qquad \frac{\Gamma\rhd M:A \qquad x:A,\Gamma\rhd N:B}{\Gamma\rhd N[x/M]:B}\ (sub)$$

$$\frac{\Gamma\rhd\lambda P:A.M:A\to B \qquad \sharp v:A\to B,\Gamma\rhd N:C}{\Gamma\rhd N[v/\lambda P:A.M]:C}\ (subf)$$

$$\frac{\Gamma\rhd M:B}{P:A,\Gamma\rhd M:B}\ (wildcard) \qquad \frac{P_1:A,P_2:A,\Gamma\rhd M:B}{@(P_1,P_2):A,\Gamma\rhd M:B}\ (layered)$$

Fig. 1. Typing rules of $TPC_{\mathrm{ES}}$.

**Remark 2.1.**

- If $\Gamma\rhd M:A$ is provable, then $FV(M)\subseteq Var(\Gamma)$.
- If $\Gamma,P:A\rhd M:B$ is provable, then $P\in AP(A)$.

Our typing rules are designed in the spirit of the Curry–Howard isomorphism by assigning patterns (resp. terms) to the types occurring in the left (resp. right)-hand side of a sequent. Therefore, the *logical* rules corresponding to our calculus are those of minimal logic that can be obtained by erasing the patterns and terms in the judgments. In particular, the logical system behind our type system contains weakening, contraction, and cut rules corresponding, respectively, to the $(wildcard)$, $(layered)$, $(let)$ and $(sub)$ typing rules. Essentially, any right sequent rule of minimal logic for a connective # corresponds to a typing rule introducing a *term* of a specific type $\tau_\#$ in

$TPC_{\mathrm{ES}}$ calculus, while any left sequent rule for a connective # corresponds to a typing rule introducing a *pattern* of type $\tau_{\#}$. Note, however, that $(+left)$ and $(\to left)$ do introduce both a pattern of type $\tau_{\#}$ and a term.

Note also that the $(app)$ typing rule, allowing to introduce a term expressing the application of a function to an argument (see Section 4.1), corresponds to a *derived* rule in the logic, obtained by applying first a left implication rule then a cut rule. Such a rule is necessary in order to close the set of typable terms under reduction, however it is not derivable from the others in the typing system. In fact, combining a $(\to left)$ and a $(let)$ rule the term $\texttt{let}\ \lambda P:A.J\ \texttt{be}\ \sharp z\ \texttt{in}\ z\ \texttt{of}\ M\ \texttt{is}\ Q\ \texttt{in}\ N$ is typable, but such a term reduces to the term $\lambda P:A.J\ \texttt{of}\ M\ \texttt{is}\ Q\ \texttt{in}\ N$ (see Section 2.3) which is not typable in the absence of the $(app)$ rule.

The cut rule itself corresponds to two distinct typing rules, introducing, respectively, let terms and explicit substitution terms. Indeed, reduction rules for let terms will allow to express the pattern matching process *inside* the $TPC_{\mathrm{ES}}$ calculus, while reduction rules for explicit substitutions terms will allow to express the behavior of explicit substitutions as is classically done in the literature [1,6,7,27]. Therefore, both pattern matching and explicit substitution computations can be modeled via the cut elimination process for sequent proofs. We will come back on this important point once the reduction rules will be presented, and we will further explain why the cut rule is interpreted via two different term constructors in the calculus.

The typing rules $(layered)$ and $(wildcard)$, corresponding, respectively, to the left structural rules contraction and weakening of minimal logic, allow to model the `as` and `wildcard` patterns largely used in programming languages such ML or Caml-Light. Therefore, we prefer to keep weakening in the logical calculus, even if it can be derived from the other logical rules we are considering. Note that the $(wildcard)$ typing rule allows us to introduce not only the wildcard pattern $\_ : A$ (as it is done in [28]), but, more generally, *any* pattern $P : A$ generated by the pattern grammar, provided that $P \in AP(A)$ and that the pattern type assignment stays linear. The role of the condition $P \in AP(A)$ is to prevent the introduction of "garbage" pattern-like expressions as, for instance, $\langle x, y\rangle : B \to C$, which would be obviously meaningless. The fact that the $(wildcard)$ introduces syntax generated by rules that are external to the typing system corresponds exactly to the behavior of the weakening rule in logic, which introduces a formula generated by an external grammar for formulae.

Note also that the formulation of the (logical) sequent rules that we have chosen is *additive* in the case of the disjunction, of the cut rule, of the right rule for conjunction and the left rule for implication, *multiplicative* for the right rule for implication and the left rule for conjunction. Such a formulation maximizes the reversibility of intuitionist rules for conjunction and disjunction, so as to simplify the definition of a type-checking algorithm (see for example [17] who gives several formulations of propositional sequent calculus).

As we observed, the $(+left)$ rule and the $(\to left)$ rule introduce both a pattern on the left-hand side of a typing judgment and a term on the right-hand side. Also, they introduce, respectively, a fresh sum variable $\xi$ and a fresh application variable $z$, whose role is to establish a link between the two sides of the judgment, which explains why both sum variables and application variables are called "communication variables".

We point out that the condition on free communication variables for the typing rules $(\rightarrow left)$, $(let)$, $(sub)$ and $(app)$, although it may seem restrictive, does not prevent the full encoding of the usual functions defined by pattern matching which are used in functional programming. Moreover, simply typed terms can be encoded in our calculus $TPC_{\mathrm{ES}}$, as shown in Section 4. In the absence of such a condition, the set of typable terms would not be closed under reduction. Indeed, the following typing judgment:

$$(y\ |_\xi\ v) : A + A \triangleright \texttt{let}\ [y\ |_\xi\ v]\ \texttt{be}\ x : A\ \texttt{in}\ [x\ |_\xi\ x] : A$$

would be provable, while the judgment

$$(y\ |_\xi\ v) : A + A \triangleright [[y\ |_\xi\ v]\ |_\xi\ [y\ |_\xi\ v]] : A,$$

where the term $[[y\ |_\xi\ v]\ |_\xi\ [y\ |_\xi\ v]]$ is obtained by reduction (see Section 2.3) of the corresponding term in the first judgment, would not be so. The condition required in rules $(\rightarrow left)$, $(let)$, $(sub)$ and $(app)$, even if omitted in [28], is also essential to recover the subject reduction property in a framework with meta-level substitutions.

In the axioms of the typing systems that we propose, the formulae $A_i$ are not necessarily atomic. One might wonder which consequences might have to restrict axioms to atomic formulae, as it is usually done, for logical systems, in the context of automated deduction. Such a restriction would have as an effect that a term like $\texttt{let}\ x\ \texttt{be}\ \langle y, v\rangle : A \times B\ \texttt{in}\ v$, for instance, would not be typable. This kind of term corresponds to an acceptable construct in a functional program: it corresponds to a case of matching of a pattern of a product type with a non-sufficiently instantiated argument, which can further be instantiated during the execution of the program.

Thus, since our primary concern in the present work is providing a logical model of pattern-matching in functional languages, the restriction of axioms to atomic formulae is indeed unwelcome. Moreover, in the typing rules of standard typed $\lambda$-calculus, the axiom rule can be used on non-atomic formulae, and one of our aims is to be able to code $\lambda$-calculus in $TPC_{\mathrm{ES}}$ (see Section 4.1).

Finally, we note that here, as opposed to the simply typed lambda calculus, a typing judgment may have several derivations. However, as we will see in what follows, one can always chose a strategy which only builds canonical derivations in order to define correct, complete and terminating type checking algorithms.

All along this paper the type decorations may be omitted from the syntax to avoid cluttering the notation. Also, by an abuse of language we will write $\Gamma \triangleright M : A$ to say that the judgment $\Gamma \triangleright M : A$ is provable.

### 2.3. Reduction rules

In what follows we use $M[\theta]$ to denote $M[x/N]$. The reduction rules of $TPC_{\mathrm{ES}}$ appear in Fig. 2.

**Definition 2.2** (Sum replacement). The replacement of a sum communication variable $\xi$ by a constant $\mathtt{K} \in \{\mathtt{L}, \mathtt{R}\}$ in a term is defined by induction on terms, modulo $\alpha$

$$
\begin{array}{lll}
(\lambda P.J)\ \texttt{of}\ M\ \texttt{is}\ Q\ \texttt{in}\ N & \longrightarrow_{Of} & \texttt{let}\ (\texttt{let}\ M\ \texttt{be}\ P\ \texttt{in}\ J)\ \texttt{be}\ Q\ \texttt{in}\ N \\
\texttt{let}\ \langle M_1, M_2\rangle\ \texttt{be}\ \langle P_1, P_2\rangle\ \texttt{in}\ N & \longrightarrow_{Pair} & \texttt{let}\ M_1\ \texttt{be}\ P_1\ \texttt{in}\ (\texttt{let}\ M_2\ \texttt{be}\ P_2\ \texttt{in}\ N) \\
\texttt{let}\ M\ \texttt{be}\ @(P_1, P_2)\ \texttt{in}\ N & \longrightarrow_{Cont} & \texttt{let}\ M\ \texttt{be}\ P_1\ \texttt{in}\ (\texttt{let}\ M\ \texttt{be}\ P_2\ \texttt{in}\ N) \\
\texttt{let}\ \texttt{inl}(L)\ \texttt{be}\ (P\ |_\xi\ Q)\ \texttt{in}\ M & \longrightarrow_{Case1} & \texttt{let}\ L\ \texttt{be}\ P\ \texttt{in}\ M\{\xi \leftarrow \texttt{L}\} \\
\texttt{let}\ \texttt{inr}(L)\ \texttt{be}\ (P\ |_\xi\ Q)\ \texttt{in}\ M & \longrightarrow_{Case2} & \texttt{let}\ L\ \texttt{be}\ Q\ \texttt{in}\ M\{\xi \leftarrow \texttt{R}\} \\
\texttt{let}\ (\lambda P.M)\ \texttt{be}\ \sharp z\ \texttt{in}\ N & \longrightarrow_{Lambda} & N[z/\lambda P.M] \\
\texttt{let}\ M\ \texttt{be}\ x\ \texttt{in}\ N & \longrightarrow_{Sub} & N[x/M] \\
\texttt{let}\ M\ \texttt{be}\ \_\ \texttt{in}\ N & \longrightarrow_{Weak} & N \\
x[x/M] & \longrightarrow_{Var1} & M \\
y[x/M] & \longrightarrow_{Var2} & y \\
(\texttt{let}\ M\ \texttt{be}\ P\ \texttt{in}\ N)[\theta] & \longrightarrow_{Dcut} & \texttt{let}\ M[\theta]\ \texttt{be}\ P\ \texttt{in}\ N[\theta] \\
\langle M, N\rangle[\theta] & \longrightarrow_{Dpair} & \langle M[\theta], N[\theta]\rangle \\
(\lambda P.M)[\theta] & \longrightarrow_{Dlambda} & \lambda P.M[\theta] \\
(z\ \texttt{of}\ M\ \texttt{is}\ P\ \texttt{in}\ N)[z/L] & \longrightarrow_{Dapp1} & L\ \texttt{of}\ M\ \texttt{is}\ P\ \texttt{in}\ N \\
(z\ \texttt{of}\ M\ \texttt{is}\ P\ \texttt{in}\ N)[x/L] & \longrightarrow_{Dapp2} & z\ \texttt{of}\ M[x/L]\ \texttt{is}\ P\ \texttt{in}\ N[x/L] \\
(\lambda Q.J\ \texttt{of}\ M\ \texttt{is}\ P\ \texttt{in}\ N)[\theta] & \longrightarrow_{Dapp3} & \lambda Q.J[\theta]\ \texttt{of}\ M[\theta]\ \texttt{is}\ P\ \texttt{in}\ N[\theta] \\
\texttt{inl}(L)[\theta] & \longrightarrow_{Dinl} & \texttt{inl}(L[\theta]) \\
\texttt{inr}(L)[\theta] & \longrightarrow_{Dinr} & \texttt{inr}(L[\theta]) \\
[M\ |_\xi\ N][\theta] & \longrightarrow_{Dcase} & [M[\theta]\ |_\xi\ N[\theta]]
\end{array}
$$

Fig. 2. Reduction rules of $TPC_{\mathrm{ES}}$.

conversion, as follows:

$$
\begin{array}{lll}
x\{\xi \leftarrow \texttt{K}\} & = & x \\
\langle M, N\rangle\{\xi \leftarrow \texttt{K}\} & = & \langle M\{\xi \leftarrow \texttt{K}\}, N\{\xi \leftarrow \texttt{K}\}\rangle \\
\texttt{inl}_A(M)\{\xi \leftarrow \texttt{K}\} & = & \texttt{inl}_A(M\{\xi \leftarrow \texttt{K}\}) \\
\texttt{inr}_A(M)\{\xi \leftarrow \texttt{K}\} & = & \texttt{inr}_A(M\{\xi \leftarrow \texttt{K}\}) \\
M[x/N]\{\xi \leftarrow \texttt{K}\} & = & M\{\xi \leftarrow \texttt{K}\}[x/N\{\xi \leftarrow \texttt{K}\}] \\
(\lambda P.M)\{\xi \leftarrow \texttt{K}\} & = & \lambda P.(M\{\xi \leftarrow \texttt{K}\}) \\
[M\ |_\xi\ N]\{\xi \leftarrow \texttt{L}\} & = & M\{\xi \leftarrow \texttt{L}\} \\
[M\ |_\xi\ N]\{\xi \leftarrow \texttt{R}\} & = & N\{\xi \leftarrow \texttt{R}\} \\
[M\ |_\rho\ N]\{\xi \leftarrow \texttt{K}\} & = & [M\{\xi \leftarrow \texttt{K}\}\ |_\rho\ N\{\xi \leftarrow \texttt{K}\}],\ \text{if}\ \xi \neq \rho \\
(\texttt{let}\ N\ \texttt{be}\ P\ \texttt{in}\ M)\{\xi \leftarrow \texttt{K}\} & = & \texttt{let}\ N\{\xi \leftarrow \texttt{K}\}\ \texttt{be}\ P\ \texttt{in}\ M\{\xi \leftarrow \texttt{K}\} \\
(z\ \texttt{of}\ N\ \texttt{is}\ Q\ \texttt{in}\ M)\{\xi \leftarrow \texttt{K}\} & = & z\ \texttt{of}\ N\{\xi \leftarrow \texttt{K}\}\ \texttt{is}\ Q\ \texttt{in}\ M\{\xi \leftarrow \texttt{K}\} \\
(\lambda P.J\ \texttt{of}\ N\ \texttt{is}\ Q\ \texttt{in}\ M)\{\xi \leftarrow \texttt{K}\} & = & \lambda P.J\{\xi \leftarrow \texttt{K}\}\ \texttt{of}\ N\{\xi \leftarrow \texttt{K}\}\ \texttt{is}\ Q\ \texttt{in}\ M\{\xi \leftarrow \texttt{K}\}
\end{array}
$$

Note that the behavior of $\{\xi \leftarrow \texttt{K}\}$ on case terms is not exactly substitution.

**Remark 2.2.** Note that, as a consequence of both the definitions of the typing rules and the reduction rules, we have the following property:

1. Each sub-term of $M$ having the form $M_1\ \texttt{of}\ M_2\ \texttt{is}\ P\ \texttt{in}\ N$ is such that $M_1$ is either an application variable or a lambda abstraction.
2. Each sub-term of $M$ having the form $M_1[z/N]$ where $z$ is an application variable is such that $N$ is a lambda abstraction.

The subsystem of $TPC_{\mathrm{ES}}$ reduction rules containing only: $Var1$, $Var2$, $Dcut$, $Dpair$, $Dlambda$, $Dapp1$, $Dapp2$, $Dapp3$, $Dinl$, $Dinr$, $Dcase$, is called ES.

**Definition 2.3** (Reduction relation $TPC_{\mathrm{ES}}$). The reduction relation $TPC_{\mathrm{ES}}$ is the reduction relation generated by the reduction rules in Fig. 2, that is, the closure of the relation generated by the reduction rules for all contexts.

We use $\longrightarrow_{TPC_{\mathrm{ES}}}$ to denote the reduction relation generated by the system $TPC_{\mathrm{ES}}$. The notations $\longrightarrow^{+}_{TPC_{\mathrm{ES}}}$ and $\longrightarrow^{*}_{TPC_{\mathrm{ES}}}$ are used to denote, respectively, the transitive and reflexive-transitive closures of $\longrightarrow_{TPC_{\mathrm{ES}}}$.

The rules in $TPC_{\mathrm{ES}}\backslash ES$ handle pattern matching as cut elimination, while the ES rules handle the explicit substitution, either by distributing it over a term $N$ (this is the case of the $D$ rules), or by actually performing it when $N$ is a variable (this is the case of the $Var1$ and $Var2$ rules). It is apparent that while the first group of rules "decomposes" a *complex* pattern $P : A$ and the corresponding term $M$, the ES rules execute the substitution of a *variable* pattern (or a sum communication variable) in a term $N$, the *Sub* rule making the bridge between the two groups. This is the reason why the *cut* sequent rule is associated to two distinct term constructors.

Note that *cut elimination steps simulate pattern matching*. Consider the $\longrightarrow_{Pair}$ reduction rule, for instance. The correspondence between such a rule and the cut elimination step in sequent proofs replacing a cut over $A_1 \wedge A_2$ with two cuts over the sub-formulae $A_1$ and $A_2$ is quite apparent. This corresponds exactly to a "principal–principal" cut case, called also "key case".

Also the ES reduction rules are modeled on cut elimination steps. For instance, the $Var1$ reduction rule corresponds to the elimination of the last cut in a sequent proof coming from an axiom (see Section 1) and a similar correspondence can be established for all the other rules of ES. This fact is completely reflected in the full proof of subject reduction given in Section 3.

This shows that cut elimination provides an enlightening rational reconstruction of computation mechanisms, such as pattern matching and explicit substitution, which are at the heart of the implementation of functional languages.

Below, we introduce some notions and lemmas that will be useful in the sequel, in order to prove properties of the $TPC_{\mathrm{ES}}$ calculus.

**Remark 2.3.** If $M \longrightarrow_{TPC_{\mathrm{ES}}} N$, then $FV(N) \subseteq FV(M)$.

**Lemma 2.4.** *The system ES is terminating.*

**Proof.** Using RPO [14,26] where symbols are ordered by the following precedence:

$$\_[\_] \succ \_ \texttt{ of } \_ \texttt{ is } \_ \texttt{ in } \_, \texttt{inl}(\_), \texttt{inr}(\_), [\_ \mid \_], \lambda\_.\_, \texttt{let } \_ \texttt{ be } \_ \texttt{ in } \_, \langle \_, \_ \rangle \qquad \square$$

**Lemma 2.5.** *The system ES is confluent.*

**Proof.** We can easily show local confluence of ES by a simple inspection of the rules which are orthogonal. Then we obtain confluence of ES by Newman's Lemma (see for example [4]) and Lemma 2.4. $\square$

**Definition 2.4.** The ES-normal form of $M$, noted $ES(M)$, is the normal form of $M$ w.r.t. the calculus ES.

Since *the system ES* is confluent, we have the following result:

**Corollary 2.6.** *ES-normal forms are unique.*

**Lemma 2.7.** *Let $M$ be a term in $\mathcal{T}_{TPC_{\mathrm{ES}}}$. Then $ES(M[x/N]) = ES(M)\{x/ES(N)\}$.*

**Proof.** The proof proceeds by induction on the structure of $M$. □

## 3. Subject reduction for $TPC_{\mathrm{ES}}$

Now, we present some intermediate notions useful in order to establish the subject reduction property for $TPC_{\mathrm{ES}}$.

**Definition 3.1.** Given a pattern $P : A$, $Dec(P : A)$ is the *deconstruction* of $P$ and is defined as follows:

$$\begin{array}{ll}
Dec(\emptyset) & = \emptyset \\
Dec(_ : A) & = _ : A \\
Dec(x : A) & = x : A \\
Dec(\sharp z : A \to B) & = \sharp z : A \to B \\
Dec((P_1 \mid_\xi P_2) : A_1 + A_2) & = (P_1 \mid_\xi P_2) : A_1 + A_2 \\
Dec(\langle P_1, P_2\rangle : A_1 \times A_2) & = Dec(P_1 : A_1), Dec(P_2 : A_2) \\
Dec(@(P_1, P_2) : A) & = Dec(P_1 : A), Dec(P_2 : A)
\end{array}$$

This notion extends naturally to a pattern type assignment $\Gamma = P_1 : A_1, \ldots, P_n : A_n$ by defining $Dec(\Gamma)$ as $Dec(P_1 : A_1), \ldots, Dec(P_n : A_n)$.

It is easy to prove that typing is stable by $Dec()$. Actually a stronger property holds:

**Lemma 3.1.** *$\Gamma \triangleright M : A$ if and only if $\Gamma_1, Dec(\Gamma_2) \triangleright M : A$, for every partition of $\Gamma$ into two disjoint sets $\Gamma_1, \Gamma_2$.*

**Proof.** The proof of the *if* part is trivial since it is sufficient to apply the rules $(layered)$ and $(\times left)$ to go from $Dec(\Gamma_2)$ to $\Gamma_2$. The proof of the *only if* part can be done by induction on the height of the proof of $\Gamma \triangleright M : A$. □

In particular, we obtain as a corollary that $\Gamma \triangleright M : A$ if and only if $Dec(\Gamma) \triangleright M : A$.

Given two typing rules $R$ and $W$, it is not true in general that they permute. Consider for example a proof ending with an application of (+ left) followed by (layered):

$$\dfrac{\dfrac{P : A, R : A + B \triangleright M : C \qquad Q : B, R : A + B \triangleright N : C}{(P \mid_\xi Q) : A + B, R : A + B \triangleright [M \mid_\xi N] : C}\,(+left)}{@((P \mid_\xi Q), R) : A + B \triangleright [M \mid_\xi N] : C}\,(layered)$$

It is easy to see that there is no proof of the typing judgment $@((P \mid_\xi Q), R) : A + B \triangleright [M \mid_\xi N] : C$ ending with an application of (layered) followed by (+ left). However, there are several cases where this permutation property does hold as shown by the following lemma.

**Lemma 3.2** (Permutations). (1) *Let $R$ be any rule in $\{\rightarrow left, +left, app, \times right, +right1, +right2, \rightarrow right, let, sub*\}$. If $\mathscr{P}$ is a proof of $\Gamma \triangleright M : A$ ending by an application of $R$ and then wildcard, there is a proof $\mathscr{P}'$ of the same judgment $\Gamma \triangleright M : A$ ending with wildcard and then $R$.*

(2) *Let $W$ be a rule in $\{\times left, layered\}$ and let $R = \{app, \times right, +right1, +right2, \rightarrow right, let, sub*\}$. If $\mathscr{P}$ is a proof of $\Gamma \triangleright M : A$ ending by an application of $R$ and then $W$, there is a proof $\mathscr{P}'$ of the same judgment $\Gamma \triangleright M : A$ ending with $W$ and then $R$.*

(3) *Let $W_1$ and $W_2$ be rules in $\{wildcard, \times left, layered\}$. If $\mathscr{P}$ is a proof of $\Gamma \triangleright M : A$ ending with an application of $W_1$ and then $W_2$ such that both rules introduce disjoint patterns, then there is a proof $\mathscr{P}'$ of the same judgment $\Gamma \triangleright M : A$ ending with $W_2$ and then $W_1$.*

**Proof.** By cases. □

In order to prove the subject reduction property we will simply reason by induction on terms. As remarked in Section 2.2 there may be several derivations for a given typing judgment $\Gamma \triangleright M : A$. However, Lemma 3.3 will allow us to restrict our attention to a particular derivation $\mathscr{D}$ of the judgment $\Gamma \triangleright M : A$ ending with an *introduction* of the term $M$. As a consequence, we will find among the sub-derivations of $\mathscr{D}$ those that introduce the sub-terms of $M$, and this fact will enable us to apply the induction hypothesis. Moreover, Lemma 3.3 also gives us a canonical derivation for some judgments of the form $\Gamma, P : B \triangleright M : A$.

**Lemma 3.3** (Introductions). (1) *Let $M$ be a term other than a variable, a case or a variable application. If $\mathscr{P}$ is a proof of $\Gamma \triangleright M : A$, then there is a proof $\mathscr{P}'$ of the same judgment ending with a rule $R_M$ which introduces the term $M$.*

(2) *Let $M$ be a variable application or a case. If $\mathscr{P}$ is a proof of $\Gamma \triangleright M : A$, then there is a proof of $Dec(\Gamma) \triangleright M : A$ ending with a rule $R_M$ which introduces the term $M$.*

(3) *If there is a proof $\mathscr{P}$ of $\Gamma, P : A \triangleright N : B$, where $P$ is either a layered pattern, a product pattern, or a wildcard pattern $\_$, then there is a proof $\mathscr{P}'$ of the same judgment such that the last inference rule introduces the pattern $P : A$.*

**Proof.** We prove the three statements by induction on the height of the proof $\mathscr{P}$. If $h = 0$, then $\Gamma \triangleright M : A$ is an axiom and all the properties hold vacuously. Let us suppose that $h > 0$.

1. We consider all the possible cases for the last rule $R$ used in $\mathscr{P}$.
   - If the last rule $R$ is $\times right$, $+right1$, $+right2$, $\rightarrow right$, $let$, $sub*$ or $app$, then we can take $\mathscr{P}' = \mathscr{P}$.

- If the last rule is $\rightarrow left$ (resp. $+left$), then the statement vacuously holds.
- If the last rule is *wildcard*, then we have a proof of $\Gamma', P:B \triangleright M:A$ obtained from a proof of $\Gamma' \triangleright M:A$ by an application of *wildcard*. We know that the statement holds for $\Gamma' \triangleright M:A$ by induction hypothesis, so it is sufficient to push upwards the *wildcard* rule using Lemma 3.2 (item 1).
- If the last rule is $\times left$ or *layered*, then the result holds by the induction hypothesis and the possibility of pushing $\times left$ and *layered* upwards via Lemma 3.2 (item 2).

2. We consider all the possible cases for the last rule $R$ used in $\mathscr{P}$.
   - If the last rule $R$ is $\times right$, $+right1$, $+right2$, $\rightarrow right$, *let*, *sub*$*$ or $app$, then the statement vacuously holds.
   - If the last rule is $\rightarrow left$ (resp. $+left$), then we know by Lemma 3.1 that there is a proof of $Dec(\Gamma) \triangleright M:A$. Since $Dec(\Gamma)$ has no product and no layered patterns, then the proof of $Dec(\Gamma) \triangleright M:A$ ends either with $\rightarrow left$ (resp. $+left$), in which case we are done, or with a *wildcard* rule, in which case we apply Lemma 3.2 (item 1) to push the *wildcard* rule upwards.
   - If the last rule is *wildcard*, then we have a proof of $\Gamma', P:B \triangleright M:A$ obtained by a *wildcard* rule from a proof of $\Gamma' \triangleright M:A$. We know that the statement holds for $\Gamma' \triangleright M:A$ by induction hypothesis so that there is a proof $\mathscr{P}'$ of $Dec(\Gamma') \triangleright M:A$ ending with a rule $R_M$ which introduces $M$. Now, let us suppose that $Dec(P:B) = P_1:B_1, \ldots, P_n:B_n$. Then $Dec(\Gamma'), Dec(P:B) \triangleright M:A$ is provable from $\mathscr{P}'$ by $n$ application of the *wildcard* rule, and we can push upwards these rules by application of Lemma 3.2 (item 1) so that we obtain the desired result.
   - If the last rule is $\times left$ or *layered*, then the result just holds by the induction hypothesis and the definition of $Dec()$.

3. To show this property, suppose the last rule used in $\mathscr{P}$ introduces the pattern $P:A$. Then we are done. Otherwise, if $P$ is $\_$, then we can show the property by induction on the height of the proof $\mathscr{P}$. The last case is when $P$ is either a product pattern $\langle P_1, P_2 \rangle$ or a layered pattern $@(P_1, P_2)$. Then we know by the proof of Lemma 3.1 that there is a proof of $\Gamma, Dec(P_1):A_1, Dec(P_2):A_2 \triangleright N:B$, where $A_1 = A_2 = A$ if $P$ is a layered pattern and $A = A_1 \times A_2$ if $P$ is a product pattern. We can now, also by Lemma 3.1, reconstruct a proof of $\Gamma, P_1:A_1, P_2:A_2 \triangleright N:B$, and thus we apply either the $(\times left)$ or $(layered)$ rule to obtain $P$ as follows:

$$\frac{\Gamma, P_1:A_1, P_2:A_2 \triangleright N:B}{\Gamma, P:A \triangleright N:B}$$ □

In order to prove the subject-reduction property we still need the following preliminary results:

**Lemma 3.4.** *If* $P \in AP(A)$, *then* $P_{(\xi,\mathrm{K})} \in AP(C)$ *for some type* $C$.

**Proof.** By induction on patterns. □

**Lemma 3.5.** *If* $\Gamma \rhd M:A$, *then* $\Gamma_{(\xi,\mathtt{K})} \rhd M\{\xi \leftarrow \mathtt{K}\}:A$.

**Proof.** See the appendix. □

**Lemma 3.6** (Loosening). *Let* $\Gamma, P:A \rhd M:B$ *be a provable judgment. We have*:
1. *If* $P=(P_1 \mid_\xi P_2)$ *and* $\xi \notin FV(M)$ *then* $\Gamma \rhd M:B$ *is provable.*
2. *If* $Var(P)\cap FV(M)=\emptyset$, *then* $\Gamma \rhd M:B$ *is provable.*

**Proof.** See the appendix. □

**Theorem 3.7** ($TPC_{\mathrm{ES}}$ subject-reduction). *If* $\Gamma \rhd T:D$ *and* $T \longrightarrow_{TPC_{\mathrm{ES}}} T'$, *then* $\Gamma \rhd T':D$.

**Proof.** By induction on $T$.

*Base case*: $T$ is a variable $x$. In this case, $T$ is in normal form, so there is nothing to show.

*Inductive step*: We first remark that in all the cases where $T \longrightarrow_{TPC_{\mathrm{ES}}} T'$ is an internal reduction step, the property immediately follows using Lemma 3.3 (items 1 and 2) and the induction hypothesis, so we are left to show the property for the cases where reduction takes place at the head of the term $T$.

1. If $T \equiv \langle M,N\rangle$, $T \equiv \lambda P : A.M$, $T \equiv \mathtt{inl}_B(M)$, $T \equiv \mathtt{inr}_B(M)$, $T \equiv z$ `of` $M$ `is` $P : A$ `in` $M$, $T \equiv [M \mid_\xi N]$, head reductions are not possible.
2. $T \equiv \lambda Q:A.M$ `of` $N$ `is` $P:B$ `in` $L$. This term is head reducible by an $Of$-rule. By Lemma 3.3, item 1, we have a proof ending in

$$\dfrac{\dfrac{\Gamma, Q:A \rhd M:B}{\Gamma \rhd \lambda Q:A.M:A\to B}\,(\to right) \qquad \Gamma \rhd N:A \qquad \Gamma, P:B \rhd L:D}{\Gamma \rhd \lambda Q:A.M \ \mathtt{of}\ N\ \mathtt{is}\ P:B\ \mathtt{in}\ L:D}\,(app)$$

where $FCV(\lambda Q:A.M)=FCV(N)=\emptyset$.

Then we have a proof ending in

$$\dfrac{\dfrac{\Gamma \rhd N:A \qquad \Gamma, Q:A \rhd M:B}{\Gamma \rhd \mathtt{let}\ N\ \mathtt{be}\ Q:A\ \mathtt{in}\ M:B}\,(let) \qquad \Gamma, P:B \rhd L:D}{\Gamma \rhd \mathtt{let}\ (\mathtt{let}\ N\ \mathtt{be}\ Q:A\ \mathtt{in}\ M)\ \mathtt{be}\ P:B\ \mathtt{in}\ L:D}\,(let)$$

We remark that since $FCV(N)=\emptyset$, then we can apply the first *let* rule and, since $FCV(\mathtt{let}\ N\ \mathtt{be}\ Q:A\ \mathtt{in}\ M)=FCV(N)\cup(FCV(M)\setminus Var(Q))=FCV(N)\cup FCV(\lambda Q:A.M)=\emptyset$ the second *let* rule is also applicable.

3. $T \equiv \mathtt{let}\ M\ \mathtt{be}\ P:A\ \mathtt{in}\ N$. By hypothesis there is a proof $\mathscr{P}$ of the judgment $\Gamma \rhd \mathtt{let}\ M\ \mathtt{be}\ P:A\ \mathtt{in}\ N:D$ and an application of Lemma 3.3, item 1, allows

us to suppose, without loss of generality, that $\mathscr{P}$ ends with a *let* rule. There are several sub-cases that need to be considered:

(a) $T \equiv \mathtt{let}\ \langle M_1, M_2\rangle\ \mathtt{be}\ \langle P_1, P_2\rangle : A_1 \times A_2\ \mathtt{in}\ N$, and thus

$$T' \equiv \mathtt{let}\ M_1\ \mathtt{be}\ P_1 : A_1\ \mathtt{in}\ (\mathtt{let}\ M_2\ \mathtt{be}\ P_2 : A_2\ \mathtt{in}\ N).$$

By Lemma 3.3, items 1 and 3, we can suppose, without loss of generality, that $\mathscr{P}$ ends with

$$\dfrac{\dfrac{\Gamma \triangleright M_1 : A_1 \qquad \Gamma \triangleright M_2 : A_2}{\Gamma \triangleright \langle M_1, M_2\rangle : A_1 \times A_2}(\times right) \qquad \dfrac{\Gamma, P_1 : A_1, P_2 : A_2 \triangleright N : D}{\Gamma, \langle P_1, P_2\rangle : A_1 \times A_2 \triangleright N : D}(\times left)}{\Gamma \triangleright \mathtt{let}\ \langle M_1, M_2\rangle\ \mathtt{be}\ \langle P_1, P_2\rangle : A_1 \times A_2\ \mathtt{in}\ N : D}(let)$$

where $FCV(\langle M_1, M_2\rangle) = \emptyset$. Hence we have the following proof for $T'$:

$$\dfrac{\Gamma \vdash M_1 : A_1 \qquad \dfrac{\dfrac{\Gamma \vdash M_2 : A_2}{\Gamma, P_1 : A_1 \triangleright M_2 : A_2}(wild) \qquad \Gamma, P_1 : A_1, P_2 : A_2 \triangleright N : D}{\Gamma, P_1 : A_1 \triangleright \mathtt{let}\ M_2\ \mathtt{be}\ P_2 : A_2\ \mathtt{in}\ N : D}(let)}{\Gamma \triangleright \mathtt{let}\ M_1\ \mathtt{be}\ P_1 : A_1\ \mathtt{in}\ (\mathtt{let}\ M_2\ \mathtt{be}\ P_2 : A_2\ \mathtt{in}\ N) : D}(let)$$

Note that the application of the *wildcard* typing rule is possible since $P_1 \in AP(A_1)$ and $P_1 : A_1, \Gamma$ is linear.

(b) $M \equiv \lambda Q : B.L$, $P$ is $\sharp z$, $A$ is $B \to C$, $T' \equiv N[z/\lambda Q : B.L]$, so that the reduction rule is *Lambda*. Thus, by Lemma 3.3, item 1, we can suppose that $\mathscr{P}$ ends with

$$\dfrac{\Gamma \triangleright \lambda Q : B.L : B \to C \qquad \sharp z : B \to C, \Gamma \triangleright N : D}{\Gamma \triangleright \mathtt{let}\ \lambda Q : B.L\ \mathtt{be}\ \sharp z : B \to C\ \mathtt{in}\ N : D}(let)$$

where $FCV(\lambda Q : B.L) = \emptyset$. Thus, we trivially have a proof ending with a *sub*$*$ typing rule:

$$\dfrac{\Gamma \triangleright \lambda Q : B.L : B \to C \qquad \sharp z : B \to C, \Gamma \triangleright N : D}{N[z/\lambda Q : B.L] : D}(subf)$$

Hence, the $\Lambda$ reduction rule corresponds to *no modification of the logical proof* underlying the typing proof of the reduced term.

(c) $T \equiv \mathtt{let}\ M\ \mathtt{be}\ x : A\ \mathtt{in}\ N$ and $T' \equiv N[x/M]$. This case is quite analogous to the one immediately above. Again, the *Sub* reduction rule corresponds to *no modification of the logical proof* underlying the typing proof of the reduced term.

(d) $T \equiv \texttt{let}\ M\ \texttt{be}\ \_ : A\ \texttt{in}\ N$ and $T' \equiv N$. By Lemma 3.3, items 1 and 3 we can suppose that $\mathscr{P}$ ends with

$$\dfrac{\Gamma \triangleright M : A \qquad \dfrac{\Gamma \triangleright N : D}{\_ : A, \Gamma \triangleright N : D}(wild)}{\Gamma \triangleright \texttt{let}\ M\ \texttt{be}\ \_ : A\ \texttt{in}\ N : D}(let)$$

Thus, trivially, $\Gamma \triangleright N : D$ is provable.

(e) $T \equiv \texttt{let}\ M\ \texttt{be}\ @(P_1, P_2) : A\ \texttt{in}\ N$ and $T' \equiv \texttt{let}\ M\ \texttt{be}\ P_1 : A\ \texttt{in}\ (\texttt{let}\ M\ \texttt{be}\ P_2 : AN)$. Again by Lemma 3.3, items 1 and 3 we may suppose that $\mathscr{P}$ ends with

$$\dfrac{\Gamma \triangleright M : A \qquad \dfrac{\Gamma, P_1 : A, P_2 : A \triangleright N : D}{\Gamma, @(P_1, P_2) : A \triangleright N : D}(lay)}{\Gamma \triangleright \texttt{let}\ M\ \texttt{be}\ @(P_1, P_2) : A\ \texttt{in}\ N : D}(let)$$

where $FCV(M) = \emptyset$. Hence, we can transform $\mathscr{P}$ in a proof $\mathscr{P}'$ ending in

$$\dfrac{\Gamma \triangleright M : A \qquad \dfrac{\dfrac{\Gamma \triangleright M : A}{P_1 : A, \Gamma \triangleright M : A}(wild) \qquad P_1, : A, P_2 : A, \Gamma \triangleright N : D}{P_1 : A, \Gamma \triangleright \texttt{let}\ M\ \texttt{be}\ P_2\ \texttt{in}\ N}(let)}{\Gamma \triangleright \texttt{let}\ M\ \texttt{be}\ P_1\ \texttt{in}\ (\texttt{let}\ M\ \texttt{be}\ P_2\ \texttt{in}\ N)}(let)$$

Note that $P_1 \in AP(A)$ and $P_1 : A, \Gamma$ is linear, so the *wildcard* rule can be applied. Also, the *let* rules can be applied since $FCV(M)$ is the empty set.

(f) $T \equiv \texttt{let}\ \texttt{inl}(M)\ \texttt{be}\ (P_1 \mid_\xi P_2) : B_1 + B_2\ \texttt{in}\ N$ and $T' \equiv \texttt{let}\ M\ \texttt{be}\ P_1\ \texttt{in}\ N\{\xi \leftarrow \mathtt{L}\}$, so that the used reduction rule is *Case*1. By Lemma 3.3, item 1, there is a proof ending in

$$\dfrac{\dfrac{\Gamma \triangleright M : B_1}{\Gamma \triangleright \texttt{inl}(M) : B_1 + B_2}(+right1) \qquad \Gamma, (P_1 \mid_\xi P_2) : B_1 + B_2 \triangleright N : D}{\Gamma \triangleright \texttt{let}\ \texttt{inl}(M)\ \texttt{be}\ (P_1 \mid_\xi P_2) : B_1 + B_2\ \texttt{in}\ N : D}(let)$$

To show that there is a proof of $\Gamma \triangleright T' : D$ it is sufficient to show that there is a proof of $\Gamma \triangleright M : B_1$ (which holds by hypothesis) and another one of $\Gamma, P_1 : B_1 \triangleright N\{\xi \leftarrow \}\mathtt{L} : D$. Since $(\Gamma, (P_1 \mid_\xi P_2) : B_1 + B_2)_{(\xi,\mathtt{L})} \triangleright N\{\xi \leftarrow \}\mathtt{L} : D$ is provable by Lemma 3.5, and $(\Gamma, (P_1 \mid_\xi P_2))_{(\xi,\mathtt{L})} : B_1 + B_2 = \Gamma, P_1 : B_1$, then

the property holds. The case $T \equiv \texttt{let inr}(M)\ \texttt{be}\ (P_1 \mid_\xi P_2) : B_1 + B_2\ \texttt{in}\ N$ is analogous to the one immediately above.

4. $T \equiv N[v/M]$ where $v$ is either a usual variable or an application communication variable.

In both cases, by Lemma 3.3, item 1, we may suppose to have a proof $\mathscr{P}$ ending with a *sub*$*$ rule:

$$\frac{\Gamma \triangleright M : A \qquad ?v : A, \Gamma \triangleright N : D}{\Gamma \triangleright N[v/M] : D}\ (sub*)$$

such that $FCV(M) = \emptyset$.

If $T'$ has been obtained by a head reduction of $T$, several sub-cases need to be considered. Recall that the following situations:

- $N \equiv U\ \texttt{of}\ S\ \texttt{is}\ Q\ \texttt{in}\ L$, $U$ is neither an application variable nor an abstraction term,
- $N$ is any term, $v$ is an application variable but $M$ is not an abstraction term,

need not be considered (see Remark 2.2). Thus, we are left with the following cases:

(a) $N \equiv x$, where $x$ is a usual variable and $v$ is $x$ itself.
Thus, the used reduction rule is *Var*1 and $T' \equiv M$ and $D \equiv A$. In this case, by Lemma 3.3, item 1, we can assume that the proof $\mathscr{P}$ typing $T$ ends with

$$\frac{\Gamma \triangleright M : A \qquad x : A, \Gamma \triangleright x : A}{\Gamma \triangleright x[x/M] : A}\ (sub)$$

As a consequence, $\Gamma \triangleright M : A$, i.e., $\Gamma \triangleright T' : A$, is provable.

(b) $N \equiv x$, where $x$ is a usual variable and $v$ is a distinct variable $y$.
Thus, the used reduction rule is *Var*2 and $T' \equiv x$. In this case, the proof $\mathscr{P}$ typing $T$ ends with

$$\frac{\Gamma \triangleright M : A \qquad y : A, \Gamma \triangleright x : D}{\Gamma \triangleright x[y/M] : D}\ (sub)$$

Since $y : A, \Gamma \triangleright x : D$ is provable, then by Lemma 3.6, item 2, the judgment $\Gamma \triangleright x : D$, i.e., $\Gamma \triangleright T' : D$, is provable.

(c) $N \equiv \texttt{inl}_B(L)$.
Here, the applied reduction rule is *Dinl*, the type $D$ of $T$ is $C + B$ and $T' \equiv \texttt{inl}_B(L[v/M])$. By Lemma 3.3, item 1, since $?v : A, \Gamma \triangleright \texttt{inl}_B(L) : C + B$ is typable, there is a proof ending in

$$\frac{?v : A, \Gamma \triangleright L : C}{?v : A, \Gamma \triangleright \texttt{inl}_B(L) : C + B}\ (+right1)$$

We can then construct a proof ending in

$$\dfrac{\dfrac{\Gamma \rhd M : A \qquad ?v : A, \Gamma \rhd L : C}{\Gamma \rhd L[v/M] : C}\,(sub*)}{\Gamma \rhd \mathtt{inl}_B(L[v/M]) : C + B}\,(+right1)$$

The case $\mathtt{inr}_B(L)$ is similar.

(d) $N \equiv \lambda P : B.J$. Then by Lemma 3.3, item 1, we have a proof ending in

$$\dfrac{?v : A, \Gamma, P : B \rhd J : C}{?v : A, \Gamma \rhd \lambda P : B.J : B \to C}\,(\to right)$$

We can then construct a proof ending in

$$\dfrac{\dfrac{\dfrac{\Gamma \rhd M : A}{\Gamma, P : B \rhd M : A}\,(Weak) \qquad ?v : A, \Gamma, P : B \rhd J : C}{\Gamma, P : B \rhd J[v/M] : C}\,(sub*)}{\Gamma \rhd \lambda P : B.J[v/M] : B \to C}\,(\to right)$$

We remark that $FCV(M)$ by hypothesis so that the $(sub*)$ rule is correct.

(e) $N \equiv [L \mid_\xi S]$.
Thus, $T' \equiv [L[v/M] \mid_\xi S[v/M]]$, and the applied rule is *Dcase*. Since $\Gamma, ?v : A \rhd [L \mid_\xi S] : D$ is provable, by Lemma 3.3, item 2, there is a proof of $Dec(\Gamma), ?v : A \rhd [L \mid_\xi S] : D$ ending with a rule introducing the term $[L \mid_\xi S]$. We have $Dec(\Gamma, ?v : A) = \Delta, ?v : A, (P \mid_\xi Q) : B + C$, for some $\Delta$. Hence, there is a proof ending in

$$\dfrac{\Delta, ?v : A, P : B \rhd L : D \qquad \Delta, ?v : A, Q : C \rhd S : D}{\Delta, ?v : A, (P \mid_\xi Q) : B + C \rhd [L \mid_\xi S] : D}\,(+left)$$

By Lemma 3.1, since $\Gamma \rhd M : A$ is provable, we also have a proof of $Dec(\Gamma) \rhd M : A$, so that $\Delta, (P \mid_\xi Q) : B + C \rhd M : A$ is provable. Now, since $FCV(M)$ is empty by hypothesis, then $\xi \notin FV(M)$ and by Lemma 3.6, item 2, we have that $\Delta \rhd M : A$ is provable. We can then construct the following proof:

$$\dfrac{\dfrac{\Delta, ?v : A, P : B \rhd L : D \qquad \dfrac{\Delta \rhd M : A}{\Delta, P : B \rhd M : A}}{\Delta, P : B \rhd L[v/M] : D} \qquad \dfrac{\Delta, ?v : A, Q : C \rhd S : D \qquad \dfrac{\Delta \rhd M : A}{\Delta, Q : C \rhd M : A}}{\Delta, Q : C \rhd S[v/M] : D}}{\Delta, (P \mid_\xi Q) : B + C \rhd [L[v/M] \mid_\xi S[v/M]] : D}$$

Thus, by Lemma 3.1, we have that $\Gamma \triangleright T' : D$ is provable.

(f) $N \equiv \langle T_1, T_2 \rangle$, $D$ is $B \times C$.

Again, $v$ may be either a usual variable or an application communication variable (this holds for all the following sub-cases too). Here, $T' \equiv \langle T_1[v/M], T_2[v/M] \rangle$, the type $D$ is $B \times C$ and the applied reduction rule is $D\,pair$. Since $?v : A, \Gamma \triangleright \langle T_1, T_2 \rangle : B \times C$ is provable, by Lemma 3.3, item 1, we may suppose to have a proof $\mathscr{P}$ ending in

$$\frac{?v : A, \Gamma \triangleright T_1 : B \qquad ?v : A, \Gamma \triangleright T_2 : C}{?v : A, \Gamma \triangleright \langle T_1, T_2 \rangle : B \times C} \; (\times right)$$

Thus we can construct a proof ending in

$$\frac{\dfrac{\Gamma \triangleright M : A \qquad ?v : A, \Gamma \triangleright T_1 : C}{\Gamma \triangleright T_1[v/M] : C}\,(sub*) \qquad \dfrac{\Gamma \triangleright M : A \qquad ?v : A, \Gamma \triangleright T_2 : B}{\Gamma \triangleright T_2[v/M] : B}\,(sub*)}{\Gamma \triangleright \langle T_1[v/M], T_2[v/M] \rangle : B \times C}$$

where the inference rule $(sub*)$ is either $(sub)$ or $(sub\,f)$.

(g) $N \equiv \texttt{let}\ R\ \texttt{be}\ P : B\ \texttt{in}\ L$.

Here, $T' \equiv \texttt{let}\ R[v/M]\ \texttt{be}\ P : B\ \texttt{in}\ L[v/M]$ and the applied rule is $Dcut$. By Lemma 3.3, item 1, since $?v : A, \Gamma \triangleright \texttt{let}\ R\ \texttt{be}\ P : B\ \texttt{in}\ L : D$ is provable, we have a proof $\mathscr{P}$ ending in

$$\frac{?v : A, \Gamma \triangleright R : B \qquad ?v, P : B, \Gamma \triangleright L : D}{?v : A, \Gamma \triangleright \texttt{let}\ R\ \texttt{be}\ P : B\ \texttt{in}\ L : D}\; (let)$$

where $FCV(R) = \emptyset$.

Thus, we can obtain a proof of

$$\frac{\dfrac{\Gamma \triangleright M : A \qquad ?v : A, \Gamma \triangleright R : B}{\Gamma \triangleright R[v/M] : B} \qquad \dfrac{\dfrac{\Gamma \triangleright M : A}{P : B, \Gamma \triangleright M : A}\,(wild) \qquad P : B, \Gamma, ?v : A \triangleright L : D}{P : B, \Gamma \triangleright L[v/M] : D}\,(let)}{\Gamma \triangleright \texttt{let}\ R[v/M]\ \texttt{be}\ P : B\ \texttt{in}\ L[v/M] : D}$$

where the *wildcard* rule is possible since $P \in AP(B)$ and $P : B, \Gamma$ is linear. The application of the last *let* rule is possible since $FCV(R[v/M]) = FCV(M) \cup (FCV(R) \setminus \{v\}) = \emptyset$.

(h) $N \equiv U$ `of` $S$ `is` $Q : C$ `in` $L$. The cases to be considered are those where some of the rules $Dapp1, Dapp2, Dapp3$ is applied.

We have three different cases (see Remark 2.2 in Section 2.3):

- The term $U$ is an application communication variable $z$ such that variable $z \not\equiv v$. In this case, the applied reduction rule is $Dapp2$ and $T' \equiv z$ `of` $S[v/M]$ `is` $Q : CL[v/M]$. Since

$$?v : A, \Gamma \rhd z \texttt{ of } S \texttt{ is } Q : C \texttt{ in } L : D$$

is provable, by Lemma 3.3, item 2, there is a proof $\mathscr{P}$ ending in

$$\frac{\Delta, ?v : A \rhd S : B \qquad \Delta, ?v : A, Q : C \rhd L : D}{\Delta, \sharp z : B \to C, ?v : A \rhd z \texttt{ of } S \texttt{ is } Q : C \texttt{ in } L : D}(\longrightarrow left)$$

where $Dec(\Gamma) = \Delta, \sharp z : B \to C$, and $FCV(S) = \emptyset$. Since there is also a proof of $\Gamma \rhd M : A$, then by Lemma 3.1 there is a proof of $Dec(\Gamma) \rhd M : A$, i.e., of $\Delta, \sharp z : B \to C \rhd M : A$. By hypothesis $FCV(M)$ is empty (because of the restriction on the $sub*$ rules), hence $z \notin FV(M)$ and by Lemma 3.6, item 2, we have $\Delta \rhd M : A$. We can then construct the proof:

$$\frac{\dfrac{\Delta, ?v : A \rhd S : B \qquad \Delta \rhd M : A}{\Delta \rhd S[v/M] : B} \qquad \dfrac{\Delta, ?v : A, Q : C \rhd L : D \qquad \dfrac{\Delta \rhd M : A}{\Delta, Q : C \rhd M : A}(wild)}{\Delta, Q : C \rhd L[v/M] : D}}{\Delta, \sharp z : B \to C \rhd z \texttt{ of } S[v/M] \texttt{ is } Q : C \texttt{ in } L[v/M] : D}$$

where the last inference is a $\longrightarrow left$ and the two inferences immediately above are two instances of the $sub*$ rule ($sub$ or $subf$ according to the nature of the variable $v$). Note that $FCV(S[v/M]) = (FCV(S) \setminus \{v\}) \cup FCV(M) = \emptyset$, hence the last inference in the above proof is correct, and that the displayed application of the $wildcard$ rule is also correct, since $Q \in AP(C)$ and $\Delta, Q : C$ is linear. Thus, by Lemma 3.1, we have a proof of $\Gamma \rhd T' : D$.

- If $U$ is the variable $v$, then $v$ is necessarily an application communication variable, $M$ is a $\lambda$-abstraction $\lambda P : B.J$, (see Remark 2.2 in Section 2.3), and $A$ is a functional type $B \to C$. Therefore, the applied reduction rule is $Dapp1$ and $T' \equiv \lambda P : B.J$ `of` $S$ `is` $Q : C$ `in` $L$. By Lemma 3.3, item 2, we have a proof ending in

$$\frac{Dec(\Gamma) \rhd S : B \qquad Dec(\Gamma), Q : C \rhd L : D}{Dec(\Gamma), \sharp v : B \longrightarrow C \rhd v \texttt{ of } S \texttt{ is } Q : C \texttt{ in } L : D}(\longrightarrow left)$$

and we can then construct, by using the *layered* and $\times left$ rules, a proof ending with

$$\dfrac{\Gamma \triangleright \lambda P:B.J : B \to C \qquad \dfrac{\dfrac{Dec(\Gamma) \triangleright S:B}{\Gamma \triangleright S:B} \qquad \dfrac{\dfrac{Dec(\Gamma), Q:C \triangleright L:D}{\Gamma, Q:C \triangleright L:D}}{}}{\Gamma \triangleright \lambda P:B.J \texttt{ of } S \texttt{ is } Q:C \texttt{ in } L:D}\ (app)$$

- If $U$ is a $\lambda$-abstraction $\lambda P:B.J$, then $T' \equiv \lambda P:B.J[v/M] \texttt{ of } S[v/M] \texttt{ is } Q:C \texttt{ in } L[v/M]$ and the applied reduction rule is *Dapp*3. Since $\Gamma, ?v:A \triangleright \lambda P:B.J \texttt{ of } S \texttt{ is } Q:C \texttt{ in } L:D$ is provable, by Lemmas 3.1 and 3.3 there is a proof ending in

$$\dfrac{\dfrac{Dec(\Gamma), ?v:A, P:B \triangleright J:C}{Dec(\Gamma), ?v:A \triangleright \lambda P:B.J : B \to C} \qquad Dec(\Gamma), ?v:A \triangleright S:B \qquad Dec(\Gamma), Q:C, ?v:A \triangleright L:D}{Dec(\Gamma) \triangleright \lambda P:B.J \texttt{ of } S \texttt{ is } Q:C \texttt{ in } L:D}$$

where $FCV(\lambda P:B.J) = FCV(S) = \emptyset$. Then, we can construct three proofs ending, respectively, in

$$\dfrac{\dfrac{\dfrac{Dec(\Gamma), ?v:A, P:B \triangleright J:C}{\Gamma, ?v:A, P:B \triangleright J:C} \qquad \dfrac{\Gamma \triangleright M:A}{\Gamma, P:B \triangleright M:A}\ (wild)}{\Gamma, P:B \triangleright J[v/M]:C}\ (sub*)}{\Gamma \triangleright \lambda P:B.J[v/M] : B \to C}\ (\to right)$$

$$\dfrac{\dfrac{Dec(\Gamma), ?v:A \triangleright S:B}{\Gamma, ?v:A \triangleright S:B} \qquad \Gamma \triangleright M:A}{\Gamma \triangleright S[v/M]:B}\ (sub*)$$

$$\dfrac{\dfrac{\dfrac{Dec(\Gamma), Q:C, ?v:A \triangleright L:D}{\Gamma, Q:C, ?v:A \triangleright L:D} \qquad \dfrac{\Gamma \triangleright M:A}{\Gamma, Q:C \triangleright M:A}\ (wild)}{\Gamma, Q:C \triangleright L[v/M]:D}\ (sub*)$$

where each inference $(sub*)$ is either a $(sub)$ or a $(subf)$ inference. Hence, we can obtain a proof ending in

$$\frac{\Gamma \triangleright \lambda P : B.J[v/M] : B \to C \qquad \Gamma \triangleright S[v/M] : B \qquad \Gamma, Q : C \triangleright L[v/M] : D}{\Gamma \triangleright \lambda P : B.J[v/M] \texttt{ of } S[v/M] \texttt{ is } Q \texttt{ in } L[v/M] : D} \quad (app)$$

Note that $FCV(\lambda P : B.J[v/M]) = FCV(J[v/M]) \setminus Var(P) = ((FCV(J) \setminus \{v\}) \cup FCV(M)) \setminus Var(P)$ which is the empty-set by hypothesis. Thus there is a proof of $\Gamma \triangleright T' : D$. □

## 4. Encoding functions into the $TPC_{\mathrm{ES}}$-calculus

This section shows how to encode functions into the $TPC_{\mathrm{ES}}$-calculus. More precisely, we first show that simply typed $\lambda$-calculus can be expressed within the $TPC_{\mathrm{ES}}$-calculus in a very simple way. Secondly, we show how to add recursive types (as lists and trees) in order to express standard programs which manipulate such recursive data types. In both cases, it turns out that conditions on free communication variables used in typing rules $\to left$, $sub$, $subf$, $let$ and $app$ do not limit the expressive power of the $TPC_{\mathrm{ES}}$-calculus.

An interesting remark to make here is that in functional programming the notation

```
#let rec f = function
  p1 -> l1
 |p2 -> l2;;
```

used to define a function by cases, forces the pattern `pi` to be always written on the left-hand side of its correspondent sub-program `li`. That means that there is no independence between *sum patterns* and *case terms* as they are always syntactically connected by the symbol `->`. From a logical point of view, if we consider our typing rules, that means also that in the proof associated to such a program, the rule $(+left)$ is always applied immediately before the rule $(\to right)$.

Imagine now that `l1` and `l2` share some code, and thus we can write them respectively as `C[r1]` and `C[r2]`, where $C[\,]$ denotes a context which represents this shared code. Our formalism allows us to write the function `f` as

$$\lambda(p1 \mid_\xi p2).C[r1 \mid_\xi r2]$$

where the $\xi$ communication variable is the syntactical tool used to make explicit the connection between any pattern and its correspondent sub-program, without forcing one to be dependent on the other one.

This makes $TPC_{\mathrm{ES}}$ a rather flexible formalism to write functions by cases.

### 4.1. Encoding simply typed λ-calculus into $TPC_{\mathrm{ES}}$-calculus

The simply typed lambda calculus can be defined by the typing rules in Fig. 3 and the reduction rules in Fig. 4.

$$x_1 : A_1, \ldots x_n : A_n \rhd x_i : A_i \quad (proj)$$

$$\frac{\Gamma, x : B \rhd M : A}{\Gamma \rhd \lambda x : B.M : B \to A} \ (\to i) \qquad \frac{\Gamma \rhd M : A \to B \quad \Gamma \rhd N : A}{\Gamma \rhd (MN) : B} \ (\to e)$$

$$\frac{\Gamma \rhd M : A \quad \Gamma \rhd N : B}{\Gamma \rhd \langle M, N \rangle : A \times B} \ (\times i) \qquad \frac{\Gamma \rhd M : A \times B}{\Gamma \rhd \pi_1(M) : A} \ (\times e) \qquad \frac{\Gamma \rhd M : A \times B}{\Gamma \rhd \pi_2(M) : A} \ (\times e)$$

$$\frac{\Gamma \rhd M : A}{\Gamma \rhd \mathtt{inl}_B(M) : A + B} \ (+i1) \qquad \frac{\Gamma \rhd M : A}{\Gamma \rhd \mathtt{inr}_B(M) : B + A} \ (+i2)$$

$$\frac{\Gamma \rhd L : A + B \quad x : A, \Gamma \rhd M : C \quad y : B, \Gamma \rhd N : C}{\Gamma \rhd \text{case } L \text{ of } x : A.M \mid y : B.N : C} \ (+e)$$

Fig. 3. Typing rules for the simply typed lambda calculus.

$$\begin{array}{lcl}
(\lambda x : A.M)N & \longrightarrow & M\{x \leftarrow N\} \\
\pi_1(\langle M, N \rangle) & \longrightarrow & M \\
\pi_2(\langle M, N \rangle) & \longrightarrow & N \\
\text{case } \mathtt{inl}(L) \text{ of } x : A.M \mid y : B.N & \longrightarrow & M\{x \leftarrow L\} \\
\text{case } \mathtt{inr}(L) \text{ of } x : A.M \mid y : B.N & \longrightarrow & N\{y \leftarrow L\}
\end{array}$$

Fig. 4. Reduction rules for the simply typed lambda calculus.

The simply typed lambda calculus can be immediately translated into the typed pattern calculus. The introduction rules/constructs are already there, so they have a trivial translation. Following the usual translation of natural deduction proofs into sequent proofs, the elimination rules/constructs are translated by the corresponding left rules followed by a *let* rule.

$$(\times elim1) \quad \frac{\Gamma \rhd M : A \times B}{\Gamma \rhd \pi_1(M) : A} \quad \longmapsto$$

$$\frac{\Gamma \rhd M : A \times B \qquad \dfrac{x : A, y : B, \Gamma \rhd x : A}{\langle x, y \rangle : A \times B, \Gamma \rhd x : A} \ (\times left)}{\Gamma \rhd \mathtt{let}\ M\ \mathtt{be}\ \langle x, y \rangle : A \times B\ \mathtt{in}\ x : A} \ (let)$$

where $x, y$ are fresh usual variables. Similarly for $(\times elim2)$.

$$(\to elim)\quad \frac{\Gamma \rhd M : A \to B \qquad \Gamma \rhd N : A}{\Gamma \rhd (MN) : B} \quad \longmapsto$$

$$\frac{\Gamma \rhd M : A \to B \qquad \dfrac{\Gamma \rhd N : A \qquad y : B, \Gamma \rhd y : B}{\sharp z : A \to B, \Gamma \rhd z \texttt{ of } N \texttt{ is } y : B \texttt{ in } y : B}\ (\to left)}{\Gamma \rhd \texttt{let } M \texttt{ be } \sharp z : A \to B \texttt{ in } (z \texttt{ of } N \texttt{ is } y : B \texttt{ in } y) : B}\ (let)$$

where $y, z$ are fresh usual variables.

$$(+elim)\quad \frac{\Gamma \rhd L : A + B \qquad x : A, \Gamma \rhd M : C \qquad y : B, \Gamma \rhd N : C}{\Gamma \rhd \text{case } L \text{ of } x : A.M \mid y : B.N : C} \quad \longmapsto$$

$$\frac{\Gamma \rhd L : A + B \qquad \dfrac{x : A, \Gamma \rhd M : C \qquad y : B, \Gamma \rhd N : C}{(x \mid_\xi y) : A + B, \Gamma \rhd [M \mid_\xi N] : C}\ (+left)}{\Gamma \rhd \texttt{let } L \texttt{ be } (x \mid_\xi y) : A + B \texttt{ in } [M \mid_\xi N] : C}\ (let)$$

where $\xi$ is a fresh sum variable.

Let $M$ be a simply typed term. We denote by $M^*$ the translation of $M$ in the pattern calculus, which is recursively defined by

$$\begin{array}{ll}
x^* & =_{def}\ x \\
(\lambda x : A.M)^* & =_{def}\ \lambda x : A.M^* \\
\texttt{inl}(M)^* & =_{def}\ \texttt{inl}(M^*) \\
\texttt{inr}(M)^* & =_{def}\ \texttt{inr}(M^*) \\
(\pi_i(M))^* & =_{def}\ \texttt{let } M^* \texttt{ be } \langle x_1, x_2 \rangle : A \times B \texttt{ in } x_i \\
(MN)^* & =_{def}\ \texttt{let } M^* \texttt{ be } \sharp z : A \to B \texttt{ in } (z \texttt{ of } N^* \texttt{ is } y : B \texttt{ in } y) \\
(\text{case } L \text{ of } x : A.M \mid y : B.N)^* & =_{def}\ \texttt{let } L^* \texttt{ be } (x \mid_\xi y) : A + B \texttt{ in } [M^* \mid_\xi N^*]
\end{array}$$

where $x_1, x_2, y, z, \xi$ are fresh variables. It is easy to see that the translation is type preserving (since it mirrors a translation of proofs!) and that $(M\{x \leftarrow L\})^* = M^*\{x \leftarrow L^*\}$.

The following lemma shows that the conditions on free communication variables used in typing rules $\to left$, *sub*, *subf*, *let* and *app* is not harmful, w.r.t. the aim of encoding $\lambda$-calculus in $TPC_{\mathrm{ES}}$.

**Lemma 4.1.** *Let $M$ be a simply typed term. Then $M^*$ contains no free communication variables.*

**Proof.** By induction on the structure of simply typed terms.

- If $M = x$, then the property is trivial.

- If $M = \langle N, L\rangle$, $M = \mathtt{inl}(N)$, $M = \mathtt{inr}(N)$ or $M = \lambda P.R$, then the property holds by i.h.
- If $M = \pi_i(N)$, then $M^* = \mathtt{let}\ N^*\ \mathtt{be}\ \langle x_1, x_2\rangle : A \times B\ \mathtt{in}\ x_i$. By i.h. $N^*$ does not contain free communication variables. The property holds since $x_1$ and $x_2$ are not communication variables.
- If $M = \mathtt{case}\ L\mathtt{of}\ x : A.N \mid y : B.R$, then $M^* = \mathtt{let}\ L^*\ \mathtt{be}\ (x \mid_\xi y) : A + B\ \mathtt{in}\ [N^* \mid \xi R^*]$. By i.h $N^*$ and $L^*$ do not contain free communication variables. Also, $y$ is not a communication variable and $z$ is a communication variable, but it is bound.
- If $M = \mathtt{case}\ L\ \mathtt{of}\ x : A.N \mid y : B.R$, then $M^* = \mathtt{let}\ L^*\ \mathtt{be}\ (x \mid_\xi y) : A + B\ \mathtt{in}\ [N^* \mid_\xi R^*]$. By i.h $L^*$, $N^*$ and $R^*$ do not contain free communication variables and $x, y$ are not translated as communication variables. The property holds since $\xi$ is a communication variable, but it is bound. □

Indeed, simply typed $\lambda$-calculus can be encoded in $TPC_{\mathrm{ES}}$:

**Theorem 4.2** ($TPC_{\mathrm{ES}}$ simulates the simply typed lambda calculus). *Let $M$ and $N$ be two simply typed terms. If $M \longrightarrow N$ in the simply typed $\lambda$-calculus, then we have that $M^* \longrightarrow^+_{TPC_{\mathrm{ES}}} N^*$.*

**Proof.** By induction on the structure of $M$ using the fact that we only use fresh free communication variables in the translation. □

The issue of the possibility of defining a reverse translation from $TPC_{\mathrm{ES}}$ typable terms back to $\lambda$-calculus, and to study the relation between the two translations is beyond the scope of this paper.

### 4.2. Adding recursive types to $TPC_{\mathrm{ES}}$

This section is largely inspired by Kesner et al. [28], since the techniques used here are essentially the same.

We use the standard fold/unfold [20] technique to encode recursion. In order to define recursive types, we add a *type constant* **1** and *type variables* with $X$ ranging over all the types. We then add to types, patterns and terms:

$$A ::= \cdots \mid \mathbf{1} \mid X \mid recX.A,$$
$$P ::= \cdots \mid \bigstar \mid fold(P),$$
$$M ::= \cdots \mid \bigstar \mid fold_{X.A}(M) \mid \mu x : A.M.$$

We have now to add the following typing rules to the original system in Section 2.2, where $\{\_ \leftarrow \_\}$, denotes a meta-level substitution used for types:

$$\frac{\Gamma \triangleright M : A}{\bigstar : \mathbf{1}, \Gamma \triangleright M : A} \qquad \Gamma \triangleright \bigstar : \mathbf{1} \qquad \frac{x : A, \Gamma \triangleright M : A}{\Gamma \triangleright \mu x : A.M : A}$$

$$\frac{P : A\{X \leftarrow recX.A\}, \Gamma \triangleright M : B}{fold(P) : recX.A, \Gamma \triangleright M : B} \qquad \frac{\Gamma \triangleright M : A\{X \leftarrow recX.A\}}{\Gamma \triangleright fold_{X.A}(M) : recX.A}$$

We also add the following reduction rules to the original reduction system in Section 2.3:

$$\begin{array}{lcl} \mu x.M & \longrightarrow & M[x/\mu x.M] \\ \texttt{let } \bigstar \texttt{ be } \bigstar \texttt{ in } N & \longrightarrow & N \\ \texttt{let } fold_{X.A}(M) \texttt{ be } fold(P) \texttt{ in } N & \longrightarrow & \texttt{let } M \texttt{ be } P \texttt{ in } N \end{array}$$

Now, in order to define the following function `merge`:

```
#let rec merge = function
  ([]       , l) -> l
 |(a::l as z,[]) -> z
 |(a::l   , b::m) -> a::(b::merge(l,m));;
```

we use the following abbreviations:

$$\text{Terms } \begin{cases} nil & =_{def}\ fold_{X.(\mathbf{1}+A\times X)}(\texttt{inl}_{A\times listA}(\bigstar)) \\ c(M,L) & =_{def}\ fold_{X.(\mathbf{1}+A\times X)}(\texttt{inr}_{\mathbf{1}}(\langle M,L\rangle)) \end{cases}$$

$$\text{Pattern } \{\ (nil \mid_\xi c(P,Q)) =_{def} fold((\bigstar \mid_\xi \langle P,Q\rangle)).$$

With these new definitions, we have the following expected derived reductions sequences in the extended system:

$$\begin{array}{lcl} \texttt{let } nil \texttt{ be } (nil \mid_\xi c(P,Q)) \texttt{ in } N & \longrightarrow^* & N\{\xi \leftarrow \mathtt{L}\} \\ \texttt{let } c(M,L) \texttt{ be } (nil \mid_\xi c(P,Q)) \texttt{ in } N & \longrightarrow^* & \texttt{let } \langle M,L\rangle \texttt{ be } \langle P,Q\rangle \texttt{ in} \\ & & N\{\xi \leftarrow \mathtt{R}\} \end{array}$$

With this syntactic sugar, we proceed to express the pattern-matching program `merge` given in the introduction. For more clarity, we write $F =_{def} M\{f \leftarrow F\}$ instead of $F =_{def} \mu f.M$ and also we omit types annotations on lambda abstraction.

$$\begin{array}{l} \texttt{merge} =_{def} \lambda\langle @(z_1,((nil \mid_\xi c(x_1,l_1)))), @(z_2,((nil \mid_\zeta c(x_2,l_2))))\rangle \\ \qquad [\ [nil \mid_\zeta z_2] \mid_\xi [z_1 \mid_\zeta c(x_1,c(x_2,merge\,\langle l_1,l_2\rangle))]\ ]. \end{array}$$

This program says that `merge` is a function taking two arguments of type *listA*, each one having the form either *nil* or $c(\_,\_)$. In order to distinguish the form of the first (resp. second) argument, the communication variable $\xi$ (resp. $\zeta$) is used. The first (resp. second) argument is also called $z_1$ (resp. $z_2$) via a layered pattern. The body of the function is expressed by the term

$$[\ [nil \mid_\zeta z_2] \mid_\xi [z_1 \mid_\zeta c(x_1,c(x_2,merge\,\langle l_1,l_2\rangle))]\ ]$$

and considers, as expected, four possible cases, corresponding to four possible instantiations of the sum variables: if the argument is $\langle nil, nil\rangle$, corresponding to $\xi = \mathtt{L}, \zeta = \mathtt{L}$, then the result is *nil*, if the argument is $\langle nil, c(x_2,l_2)\rangle$, corresponding to $\xi = \mathtt{L}, \zeta = \mathtt{R}$, then the result is $z_2$, if the argument is $\langle c(x_1,l_1), nil\rangle$, corresponding to $\xi = \mathtt{R}, \zeta = \mathtt{L}$,

the result is $z_1$, and finally if the argument is $\langle c(x_1,l_1),c(x_2,l_2)\rangle$, corresponding to $\xi=\mathtt{R},\zeta=\mathtt{R}$, then the result is $c(x_1,c(x_2,\mathtt{merge}\langle l_1,l_2\rangle))$.
The closed term `merge` type-checks since the judgment $\triangleright \mathtt{merge}: listA \times listA \to listA$ is derivable in the extended typing system including the typing rules for recursion given above.

The function `merge` has the expected operational behavior using the new reduction rules recently introduced; indeed, one can verify that:

$$\begin{array}{lll}
\mathtt{merge}\langle nil, nil\rangle & \longrightarrow^* & nil \\
\mathtt{merge}\langle nil, c(a,l)\rangle & \longrightarrow^* & c(a,l) \\
\mathtt{merge}\langle c(a,l), nil\rangle & \longrightarrow^* & c(a,l) \\
\mathtt{merge}\langle c(a,nil), c(b,nil)\rangle & \longrightarrow^* & c(a,c(b,nil))
\end{array}$$

where $\mathtt{merge}\langle x,y\rangle$ abbreviates the term $\mathtt{let\ merge\ be}\ \sharp z\ \mathtt{in}\ z\ \mathtt{of}\ \langle x,y\rangle\ \mathtt{is}\ w\ \mathtt{in}\ w$ exactly as in Section 4.1.

Other interesting examples using lists can be found in [28]. Note however that in the rest of the paper we will keep on working with a calculus with no recursion, otherwise properties such as strong normalization would not be pertinent.

## 5. The calculus *TPC*

We remark that, in a pattern calculus, both pattern matching and substitutions can be either at the meta-level or in the language itself. More precisely, if both operations are in the meta-language we get, for example, the calculus in [28], if both are in the language itself we get the $TPC_{\mathrm{ES}}$ calculus, and, finally, if pattern matching is internalized while substitution is kept as a meta-level operation we get the *TPC* calculus that we introduce in this section. It is interesting to study here such a calculus because proving some properties of *TPC* will help us to prove some properties of $TPC_{\mathrm{ES}}$ (i.e. confluence and strong normalization).

The (raw) terms of the *TPC*-calculus are (raw) $TPC_{\mathrm{ES}}$-terms without explicit substitutions. The typing rules of *TPC* are those for $TPC_{\mathrm{ES}}$ except *sub*. To avoid confusion between typing judgments in *TPC* and $TPC_{\mathrm{ES}}$, when necessary we will write $\Gamma \triangleright_{TPC} M:A$ (resp. $\Gamma \triangleright_{TPC_{\mathrm{ES}}} M:A$).

The reduction rules of the *TPC*-calculus are the same as those of $TPC_{\mathrm{ES}}\backslash ES$ but explicit substitution is replaced by a meta-level substitution as Fig. 5 shows.

As expected, the calculus *TPC*-calculus also enjoys subject reduction. This is a consequence of the subject reduction property for $TPC_{\mathrm{ES}}$ (Theorem 3.7) and Lemma 2.7 which shows how *ES* implements meta-level substitutions.

**Theorem 5.1** (*TPC*-subject reduction). *If* $\Gamma \triangleright M:A$ *and* $M \longrightarrow_{TPC} N$, *then* $\Gamma \triangleright N:A$.

The following result will be used in order to prove tha main properties of *TPC*, namely, confluence and strong normalization.

$$
\begin{array}{lll}
(\lambda P.J)\ \texttt{of}\ M\ \texttt{is}\ Q\ \texttt{in}\ N & \longrightarrow_{Of} & \texttt{let}\ (\texttt{let}\ M\ \texttt{be}\ P\ \texttt{in}\ J)\ \texttt{be}\ Q\ \texttt{in}\ N \\
\texttt{let}\ \langle M_1, M_2\rangle\ \texttt{be}\ \langle P_1, P_2\rangle\ \texttt{in}\ N & \longrightarrow_{Pair} & \texttt{let}\ M_1\ \texttt{be}\ P_1\ \texttt{in}\ (\texttt{let}\ M_2\ \texttt{be}\ P_2\ \texttt{in}\ N) \\
\texttt{let}\ M\ \texttt{be}\ @(P_1, P_2)\ \texttt{in}\ N & \longrightarrow_{Cont} & \texttt{let}\ M\ \texttt{be}\ P_1\ \texttt{in}\ (\texttt{let}\ M\ \texttt{be}\ P_2\ \texttt{in}\ N) \\
\texttt{let}\ \texttt{inl}(L)\ \texttt{be}\ (P\ |_{\xi}\ Q)\ \texttt{in}\ M & \longrightarrow_{Case1} & \texttt{let}\ L\ \texttt{be}\ P\ \texttt{in}\ M\{\xi \leftarrow \texttt{L}\} \\
\texttt{let}\ \texttt{inr}(L)\ \texttt{be}\ (P\ |_{\xi}\ Q)\ \texttt{in}\ M & \longrightarrow_{Case2} & \texttt{let}\ L\ \texttt{be}\ Q\ \texttt{in}\ M\{\xi \leftarrow \texttt{R}\} \\
\texttt{let}\ (\lambda P.M)\ \texttt{be}\ \sharp z\ \texttt{in}\ N & \longrightarrow_{Lambda} & N\{z \leftarrow \lambda P.M\} \\
\texttt{let}\ M\ \texttt{be}\ x\ \texttt{in}\ N & \longrightarrow_{Sub} & N\{x \leftarrow M\} \\
\texttt{let}\ M\ \texttt{be}\ \_\ \texttt{in}\ N & \longrightarrow_{Weak} & N
\end{array}
$$

Fig. 5. Reduction rules of *TPC*.

**Lemma 5.2** (Reduction is stable under substitution). *If* $M \longrightarrow_{TPC} M'$ *and* $N \longrightarrow_{TPC} N'$, *then*

1. $M\{x \leftarrow N\} \longrightarrow_{TPC} M'\{x \leftarrow N\}$,
2. $M\{\xi \leftarrow \texttt{K}\} \longrightarrow_{TPC} M'\{\xi \leftarrow \texttt{K}\}$ *and*
3. $M\{x \leftarrow N\} \longrightarrow^*_{TPC} M\{x \leftarrow N'\}$ ($M\{x \leftarrow N\} \longrightarrow^+_{TPC} M\{x \leftarrow N'\}$ *if* $x \in FV(M)$).

**Proof.** We show the three properties by induction on $M$:

- $M = x$. Then, the first and second properties trivially hold. For the third one, $x\{x \leftarrow N\} = N \longrightarrow^+_{TPC} N' = x\{x \leftarrow N'\}$.
- $M = y$. Then the first and second properties trivially hold. For the third one, $y\{x \leftarrow N\} = y \longrightarrow^*_{TPC} y = y\{x \leftarrow N'\}$.
- For all the other cases the properties hold by induction hypothesis using the substitution lemma [5]. □

### 5.1. Confluence for TPC

It is well known that orthogonal higher-order calculi, where abstractions are only defined on *variables*, can be proved to be confluent (see for example [36]). Now, even if *TPC* could be intuitively be seen as an "orthogonal" higher-order calculus, one cannot apply the previous mentioned techniques to prove confluence of *TPC* as abstractions are now also defined on patterns. As a consequence, in order to prove the confluence property for all raw terms of *TPC* we use a method due to Tait and Martin-Löf [5], relating the reduction relation to a parallel reduction relation $\gg$. This proof technique is quite standard and has essentially four steps:

1. Define a reflexive parallel reduction relation on terms closed for all contexts, written here $\gg$.
2. Prove that the reflexive and transitive closures of and $\gg$ are the same relation.
3. Prove that $\gg$ has the diamond property.
4. Prove that a relation is confluent if it has the diamond property.

We proceed now to define the parallel reduction $\gg$ as follows:

**Definition 5.1** (The relation $\gg$).

- $M \gg M$.

- If $M \gg M'$ and $N \gg N'$, then
  - $\texttt{let}\ M\ \texttt{be}\ \_\ \texttt{in}\ N \gg N'$ and $\texttt{let}\ M\ \texttt{be}\ x\ \texttt{in}\ N \gg N'\{x \leftarrow M'\}$.
  - If $M$ is a $\lambda$-abstraction, then $\texttt{let}\ M\ \texttt{be}\ \sharp z\ \texttt{in}\ N \gg N'\{z \leftarrow M'\}$.
  - If $M = \langle M_1, M_2\rangle$ and $M' = \langle M_1', M_2'\rangle$, then $\texttt{let}\ \langle M_1, M_2\rangle\ \texttt{be}\ \langle P_1, P_2\rangle\ \texttt{in}\ N \gg \texttt{let}\ M_1'\ \texttt{be}\ P_1\ \texttt{in}\ (\texttt{let}\ M_2'\ \texttt{be}\ P_2\ \texttt{in}\ N')$.
  - If $M = \texttt{inl}(M_1)$ and $M' = \texttt{inl}(M_1')$, then $\texttt{let}\ \texttt{inl}(M_1)\ \texttt{be}\ (P_1 \mid_\xi P_2)\ \texttt{in}\ N \gg \texttt{let}\ M_1'\ \texttt{be}\ P_1\ \texttt{in}\ N'\{\xi \leftarrow \texttt{L}\}$.
  - If $M = \texttt{inr}(M_1)$ and $M' = \texttt{inr}(M_1')$, then $\texttt{let}\ \texttt{inr}(M_1)\ \texttt{be}\ (P_1 \mid_\xi P_2)\ \texttt{in}\ N \gg \texttt{let}\ M_1'\ \texttt{be}\ P_2\ \texttt{in}\ N'\{\xi \leftarrow \texttt{R}\}$,
  - $\texttt{let}\ M\ \texttt{be}\ @(P_1, P_2)\ \texttt{in}\ N \gg \texttt{let}\ M'\ \texttt{be}\ P_1\ \texttt{in}\ (\texttt{let}\ M'\ \texttt{be}\ P_2\ \texttt{in}\ N')$.
- If $M = \lambda P.J \gg \lambda P.J' = M'$, $N \gg N'$ and $L \gg L'$, then $(\lambda P.J)\ \texttt{of}\ N\ \texttt{is}\ Q\ \texttt{in}\ L \gg \texttt{let}\ (\texttt{let}\ N'\ \texttt{be}\ P\ \texttt{in}\ J')\ \texttt{be}\ Q\ \texttt{in}\ L'$.
- If $M_1 \gg M_1', \ldots, M_n \gg M_n'$, then $f(M_1, \ldots, M_n) \gg f(M_1', \ldots, M_n')$, where $f(\_, \ldots, \_)$ is $\texttt{inl}(\_), \texttt{inr}(\_), \lambda P.\_, [\_ \mid_\xi \_], \langle \_, \_\rangle, \texttt{let}\ \_\ \texttt{be}\ P\ \texttt{in}\ \_, \_\ \texttt{of}\ \_\ \texttt{is}\ P\ \texttt{in}\ \_$.

In what follows, given a binary reduction relation $\mathscr{R}$, we will say that, $M \mathscr{R} M'$ comes from "internal reductions" iff $M = C[M_1, \ldots, M_n]$ and $M' = C[M_1', \ldots, M_n']$ for some non-empty context $C[\ ]$ and $\forall i\ M_i \mathscr{R} M_i'$.

**Lemma 5.3.** *The reflexive transitive closures of* $\gg$ *and* $\longrightarrow_{TPC}$ *are the same relation.*

**Proof.** To show the inclusion $\gg^* \subseteq \longrightarrow^*_{TPC}$, we show that $\gg \subseteq \longrightarrow^*_{TPC}$ by induction on the definition of $\gg$. See the appendix for details. □

In order to prove the diamond property of $\gg$ we need the following intermediate results.

**Lemma 5.4.** *If* $M \gg M'$ *and* $N \gg N'$, *then* $M\{x \leftarrow N\} \gg M'\{x \leftarrow N'\}$.

**Proof.** By induction on the structure of $M$ using the substitution lemma [5]. □

**Lemma 5.5.** *If* $M \gg M'$, *then there is* $N$ *such that* $M\{\xi \leftarrow \texttt{K}\} \gg N$ *and* $M'\{\xi \leftarrow \texttt{K}\} \gg N$.

**Proof.** By induction on $M$ and by cases. See the appendix for details. □

**Proposition 5.6** (Diamond property for $\gg$). *The relation* $\gg$ *satisfies the diamond property*: *for every term* $L$ *such that* $L \gg L_1$ *and* $L \gg L_2$, *there exists* $L_3$ *such that* $L_1 \gg L_3$ *and* $L_2 \gg L_3$.

**Proof.** By induction on the structure of $L$. There are two cases:

- If $L \gg L_1$ and $L \gg L_2$ come from internal reductions, then there exists a context $C[\ ]$ such that $L = C[M^1, \ldots, M^n] \gg C[M_1^1, \ldots, M_1^n] = L_1$ and $L = C[M^1, \ldots, M^n] \gg C[M_2^1, \ldots, M_2^n] = L_2$, where $M^i \gg M_1^i$ and $M^i \gg M_2^i$ for $i = 1 \ldots n$. By i.h. there are

sub-terms $M_3^1,\ldots,M_3^n$ such that $M_1^i \gg M_3^i$ and $M_2^i \gg M_3^i$ for $i=1\ldots n$. Let $L_3 = C[M_3^1,\ldots,M_3^n]$, then $L_1 \gg L_3$ and $L_2 \gg L_3$.

- If $L \gg L_1$ or $L \gg L_2$ do not come both from internal reductions, we proceed by case-analysis. We just show the most interesting case as all the others are straightforward:

  If $L = \texttt{let}\ N\ \texttt{be}\ x\ \texttt{in}\ M$, there are 2 cases:

  If $\texttt{let}\ N\ \texttt{be}\ x\ \texttt{in}\ M \gg M_2\{x \leftarrow N_2\}$, where $N \gg N_2$ and $M \gg M_2$, and $\texttt{let}\ N\ \texttt{be}\ x\ \texttt{in}\ M \gg \texttt{let}\ N_1\ \texttt{be}\ x\ \texttt{in}\ M_1$, where $N \gg N_1$ and $M \gg M_1$, then there exist by i.h $N_3$ and $M_3$ such that $N_1 \gg N_3$, $N_2 \gg N_3$, $M_1 \gg M_3$, $M_2 \gg M_3$. We close the diagram using Lemma 5.4 as follows:

  $$M_2\{x \leftarrow N_2\} \gg M_3\{x \leftarrow N_3\} \qquad \texttt{let}\ N_1\ \texttt{be}\ x\ \texttt{in}\ M_1 \gg M_3\{x \leftarrow N_3\}.$$

  If $\texttt{let}\ N\ \texttt{be}\ x\ \texttt{in}\ M \gg M_2\{x \leftarrow N_2\}$, where $N \gg N_2$ and $M \gg M_2$, and $\texttt{let}\ N\ \texttt{be}\ x\ \texttt{in}\ M \gg M_1\{x \leftarrow N_1\}$, where $N \gg N_1$ and $M \gg M_1$, then there exist by i.h $N_3$ and $M_3$ such that $N_1 \gg N_3$, $N_2 \gg N_3$, $M_1 \gg M_3$, $M_2 \gg M_3$. We close the diagram using Lemma 5.4 as follows:

  $$M_2\{x \leftarrow N_2\} \gg M_3\{x \leftarrow N_3\}, \qquad M_1\{x \leftarrow N_1\} \gg M_3\{x \leftarrow N_3\}.$$

  If $L = \texttt{let}\ \texttt{inl}(M)\ \texttt{be}\ (P\ |_\xi\ Q)\ \texttt{in}\ N$, there are also two cases:

  If $\texttt{let}\ \texttt{inl}(M)\ \texttt{be}\ (P\ |_\xi\ Q)\ \texttt{in}\ N \gg \texttt{let}\ M'\ \texttt{be}\ P\ \texttt{in}\ N\{\xi \leftarrow \texttt{L}\}$, where $M \gg M'$, and $\texttt{let}\ \texttt{inl}(M'')\ \texttt{be}\ (P|_\xi Q)\ \texttt{in}\ N \gg \texttt{let}\ \texttt{inl}(M'')\ \texttt{be}\ (P|_\xi Q)\ \texttt{in}\ N'$ where $M \gg M''$ and $N \gg N'$, then we have by i.h. $M_1$ such that $M' \gg M_1$ and $M'' \gg M_1$. Therefore,

  $$\texttt{let}\ M'\ \texttt{be}\ P\ \texttt{in}\ N\{\xi \leftarrow \texttt{L}\} \gg \texttt{let}\ M_1\ \texttt{be}\ P\ \texttt{in}\ N\{\xi \leftarrow \texttt{L}\}$$

  and

  $$\begin{aligned}&\texttt{let}\ \texttt{inl}(M'')\ \texttt{be}\ (P\ |_\xi\ Q)\ \texttt{in}\ N'\\ &\gg \texttt{let}\ \texttt{inl}(M_1)\ \texttt{be}\ (P\ |_\xi\ Q)\ \texttt{in}\ N'\\ &\gg \texttt{let}\ M_1\ \texttt{be}\ P\ \texttt{in}\ N'\{\xi \leftarrow \texttt{L}\}.\end{aligned}$$

  By Lemma 5.5 we know that there is $H$ such that $N\{\xi \leftarrow \texttt{L}\} \gg H$ and $N'\{\xi \leftarrow \texttt{L}\} \gg H$ so that $\texttt{let}\ M_1\ \texttt{be}\ P\ \texttt{in}\ N\{\xi \leftarrow \texttt{L}\} \gg \texttt{let}\ M_1\ \texttt{be}\ P\ \texttt{in}\ H$ and $\texttt{let}\ M_1\ \texttt{be}\ P\ \texttt{in}\ N'\{\xi \leftarrow \texttt{L}\} \gg \texttt{let}\ M_1\ \texttt{be}\ P\ \texttt{in}\ H$ allow to close the diagram.

  If $\texttt{let}\ \texttt{inl}(M)\ \texttt{be}\ (P\ |_\xi\ Q)\ \texttt{in}\ N \gg \texttt{let}\ M'\ \texttt{be}\ P\ \texttt{in}\ N\{\xi \leftarrow \texttt{L}\}$, where $M \gg M'$, and $\texttt{let}\ \texttt{inl}(M)\ \texttt{be}\ (P|_\xi Q)\ \texttt{in}\ N \gg \texttt{let}\ M''\ \texttt{be}\ P\ \texttt{in}\ N\{\xi \leftarrow \texttt{L}\}$, where $M \gg M''$, then we have by i.h. $M_1$ such that $M' \gg M_1$ and $M'' \gg M_1$. Therefore, $\texttt{let}\ M'\ \texttt{be}\ P\ \texttt{in}\ N\{\xi \leftarrow \texttt{L}\} \gg \texttt{let}\ M_1\ \texttt{be}\ P\ \texttt{in}\ N\{\xi \leftarrow \texttt{L}\}$ and $\texttt{let}\ M''\ \texttt{be}\ P\ \texttt{in}\ N\{\xi \leftarrow \texttt{L}\} \gg \texttt{let}\ M_1\ \texttt{be}\ P\texttt{in}\ N\{\xi \leftarrow \texttt{L}\}$ concludes this case. □

It is easy to see that if a relation $R$ satisfies the diamond property then its reflexive transitive closure $R^*$ also satisfies it [5]. Hence, by Lemma 5.15 and Proposition 5.6 we can conclude:

**Theorem 5.7** (Confluence of *TPC*). *The relation* $\longrightarrow_{TPC}$ *is confluent on raw terms, i.e., if* $M \longrightarrow^*_{TPC} N$ *and* $M \longrightarrow^*_{TPC} N'$, *then there is a term* $T$ *such that* $N \longrightarrow^*_{TPC} T$ *and* $N' \longrightarrow^*_{TPC} T$.

### 5.2. Strong normalization for TPC

In this section we show strong normalization for *TPC* adapting a proof technique based on stability [29] to the case of pattern matching. The *TPC*-calculus has some rules to express internal pattern matching which can be seen as *higher order rewriting rules*, so we borrow from [25] a trick used to show strong normalization of $\lambda$-calculus mixed with a special kind of higher order rewriting rules. We may omit types from terms to make easier the notation but no formal erasure takes place.

We first define the notion of stability for well-typed terms using the notation $MN$, exactly as in Section 4, as an abbreviation of the following term:

$$MN =_{def} \texttt{let } M \texttt{ be } \sharp z : B \to C \texttt{ in } (z \texttt{ of } N \texttt{ is } y : C \texttt{ in } y),$$

where $z, y$ are fresh variables, $M$ is of type $B \to C$ and $N$ of type $B$.

We define the *outermost constructor* of a term $MN_1 \dots N_n$ as the outermost constructor of the term $M$. Thus for example, the outermost constructor of the term

$$(\lambda P.J)N = \texttt{let } (\lambda P.J) \texttt{ be } \sharp z : B \to C \texttt{ in } (z \texttt{ of } N \texttt{ is } y : C \texttt{ in } y)$$

is $\lambda$.

**Definition 5.2** (Stable terms). A term $M$ of type $A$ is defined to be *stable* as follows:

- If $A$ is an atomic type, $M$ is stable iff it is strongly normalizing.
- If $A \equiv A_1 \times A_2$, $M$ is stable iff it is strongly normalizing and whenever $M$ reduces to $\langle M_1, M_2 \rangle$, then $M_1$ and $M_2$ are both stable.
- If $A \equiv A_1 + A_2$, $M$ is stable iff it is strongly normalizing and whenever $M$ reduces to $\texttt{inl}_{A_1+A_2}(M')$ or to $\texttt{inr}_{A_1+A_2}(M')$, then $M'$ is stable.
- If $A \equiv A_1 \to A_2$, $M$ is stable iff for every stable term $N$ of type $A_1$, $MN$ is stable.

The goal of this section is to show that every *typed TPC*-term is strongly normalizing. For that, as traditionally done, we first show that every stable term is strongly normalizing (Lemma 5.11), and that every typed term is stable (Lemma 5.20).

The main difference between this proof and other proofs of strong normalization existing in the literature, is the extension of the technique to the case of patterns, that is tackled by the notion of *set-stable sets* in Definition 5.3.

**Proposition 5.8.** *Let $k \geqslant 0$. Let $M$ be a term of type $A \to B$. The term $M$ is stable if and only if $MN_1 \dots N_k$ is stable for each sequence $N_1 \dots N_k$ of stable terms of appropriate types.*

**Proof.** By induction on $k$. □

**Remark 5.9.** When $M$ is of functional type, we can then use equivalently $k > 1$ or $k = 1$ in the characterization of stability. We will use the most suitable characterization, according to the considered case. We note a sequence $N_1 \dots N_k$ as $\bar{N}$.

The strong normalization theorem follows from the following sequence of lemmas. In these lemmas "term" means *well-typed term*, but we have omitted the pattern type assignment to simplify the notation.

**Lemma 5.10.** *If $M_1, M_2, \bar{K}$ are strongly normalizing, $Q$ is a pattern and $x$ is a variable, then the following terms are all strongly normalizing*:

$$x\bar{K} \qquad \langle M_1, M_2\rangle \qquad (x \texttt{ of } M_1 \texttt{ is } Q \texttt{ in } M_2)\bar{K}$$

$$\texttt{inl}_A(M_1) \qquad \texttt{inr}_A(M_1) \qquad [M_1 \mid_\xi M_2]\bar{K}$$

**Proof.** The argument of the proof is the same for all the cases: any generic reduction sequence starting from the given term terminates because it can only reduce the terms $\bar{K}$, $M_1$ and $M_2$. □

**Lemma 5.11.** (1) *Every stable term $M$ of type $A$ is strongly normalizing.*
(2) *A term $xK_1 \dots K_n$ of type $A$ is stable for arbitrary strongly normalizing terms $K_1 \dots K_n$, where $n \geqslant 0$.*

**Proof.** The two properties can be shown at the same time by induction on the type $A$ using when necessary Lemma 5.10. See the appendix for details. □

**Corollary 5.12.** *Every variable is stable.*

**Lemma 5.13.** *If $M$ is a stable term and $M \longrightarrow_{TPC} N$, then $N$ is stable.*

**Proof.** By induction on type $A$ of $M$, as usually done. See the appendix for details. □

**Lemma 5.14.**
- $\langle M_1, M_2\rangle$ *is stable if* $M_1$ *and* $M_2$ *are stable.*
- $\texttt{inl}(M_1)$ *and* $\texttt{inr}(M_1)$ *are stable if* $M_1$ *is stable.*
- $z \texttt{ of } M_1 \texttt{ is } Q \texttt{ in } M_2$ *is stable if* $M_1$ *and* $M_2$ *are stable.*
- $[M_1 \mid_\xi M_2]$ *is stable if* $M_1$ *and* $M_2$ *are stable.*

**Proof.** Let us suppose that $M_1$ and $M_2$ are stable. By Lemma 5.11 the terms $M_1$ and $M_2$ are strongly normalizing.
- By Lemma 5.10 $\langle M_1, M_2\rangle$ is strongly normalizing. Thus, in order to conclude that $\langle M_1, M_2\rangle$ is stable, it suffices to show that whenever $\langle M_1, M_2\rangle$ reduces to $\langle M_1', M_2'\rangle$, then $M_1'$ and $M_2'$ are stable. Now, if $\langle M_1, M_2\rangle$ reduces to $\langle M_1', M_2'\rangle$, then $M_1$ reduces to $M_1'$ and $M_2$ reduces to $M_2'$, thus, by Lemma 5.13, both $M_1'$ and $M_2'$ are stable, since $M_1$ and $M_2$ are so.
- By Lemma 5.10 $\texttt{inl}(M_1)$ and $\texttt{inr}(M_1)$ are strongly normalizing and, again, the results follow by definition using Lemma 5.13.
- By Proposition 5.8, it is sufficient to show that there exists an integer $k \geqslant 0$ such that, for any sequence $\bar{N}$ of stable terms of length $k$, $(z \texttt{ of } M_1 \texttt{ is } Q \texttt{ in } M_2)\bar{N}$ is stable.

Thus, let us take $k$ big enough to guarantee that $(z \texttt{ of } M_1 \texttt{ is } Q \texttt{ in } M_2)\bar{N}$ is not a functional type. By Lemma 5.11 we know that $\bar{N}$ are strongly normalizing, and by Lemma 5.10 we know that $(z \texttt{ of } M_1 \texttt{ is } Q \texttt{ in } M_2)\bar{N}$ is strongly normalizing. Now, $(z \texttt{ of } M_1 \texttt{ is } Q \texttt{ in } M_2)\bar{N}$ cannot reduce to a pair nor to an injection. Thus we can conclude, by definition of stability, that $(z \texttt{ of } M_1 \texttt{ is } Q \texttt{ in } M_2)\bar{N}$ is stable. As a consequence, $z \texttt{ of } M_1 \texttt{ is } Q \texttt{ in } M_2$ is stable.

- As before, since $[M_1 \mid_\xi M_2]\bar{N}$ cannot reduce to a pair nor to an injection. □

In the following definition we use the notation $M \longrightarrow_{TPC,\varepsilon} N$ to say that $M$ *head* reduces to $N$. Also, we write $[\![M,P,N]\!]$ for the following set of *TPC* terms:

$$\{T \mid \texttt{let } M \texttt{ be } P \texttt{ in } N \longrightarrow^*_{TPC} \longrightarrow_{TPC,\varepsilon} \longrightarrow^*_{TPC} T\}.$$

Intuitively, $T$ is in $[\![M,P,N]\!]$ if $T$ is a reduct of $\texttt{let } M \texttt{ be } P \texttt{ in } N$ coming from a reduction sequence which removes at some point the outermost `let be in`.

**Definition 5.3** (Set-stable sets). The set $[\![M,P,N]\!]$ is said to be *set-stable* if and only if:

1. Every term in $[\![M,P,N]\!]$ is stable, and
2. $M$ and $N$ are strongly normalizing.

Note that this notion is different from that in [28]: on one hand we have to handle explicit pattern matching, and on the other one, the case $P = @(P_1,P_2)$ is not treated independently.

We are now ready to state some preliminary properties which are needed the prove the main Lemma of this section, namely, stability of typed terms (Lemma 5.20).

**Lemma 5.15.** *If* $[\![M,P,N]\!]$ *is set-stable, then* $(\texttt{let } M \texttt{ be } P \texttt{ in } N)$ *is stable.*

**Proof.** Let $\bar{K}$ be stable terms such that $L = (\texttt{let } M \texttt{ be } P \texttt{ in } N)\bar{K}$ is not of functional type. As in the previous proofs, the same reasoning handles the cases where the type of $\texttt{let } M \texttt{ be } P \texttt{ in } N$ is functional or not, by taking $\bar{K}$ to be empty in the second case. This technique will also be implicitly used in the sequel. If we show that the term $L$ is stable, then the lemma follows from Proposition 5.8.

For that, let us first show that $L$ is strongly normalizing. Consider any reduction sequence starting at $L$.

- If the outermost constructor *let* is never removed, then reductions proceed only inside $M$, $N$ and $\bar{K}$. The terms $M$ and $N$ are strongly normalizing by definition and the terms $\bar{K}$ are stable by hypothesis, so they are strongly normalizing by Lemma 5.11. As a consequence, the reduction sequence terminates.
- If the outermost constructor *let* is removed, then the reduction sequence looks like

$$L = (\texttt{let } M \texttt{ be } P \texttt{ in } N)\bar{K} \longrightarrow^*_{TPC} (\texttt{let } M' \texttt{ be } P \texttt{ in } N')\bar{K}' \longrightarrow_{TPC} T\bar{K}' \longrightarrow_{TPC} \ldots,$$

where $M \longrightarrow^*_{TPC} M'$, $N \longrightarrow^*_{TPC} N'$, $\bar{K} \longrightarrow^*_{TPC} \bar{K}'$ and $T \in [\![M,P,N]\!]$.

Since $T$ belongs to $[\![M,P,N]\!]$, it is stable by hypothesis and so $T\bar{K'}$ is stable by Proposition 5.8 and strongly normalizing by Lemma 5.11. As a consequence, such a reduction sequence also terminates.

To finish the proof suppose that $L$ reduces to a pair $\langle L_1, L_2\rangle$ or to $\mathtt{inl}(L_1)$ or to $\mathtt{inr}(L_1)$. Then, we have necessarily removed the outermost constructor $\mathtt{let}\ \_\ \mathtt{be}\ \_\ \mathtt{in}\ \_$ and we obtain a reduction sequence similar to the last one but with more steps leading to the term $\langle L_1, L_2\rangle$, $\mathtt{inl}(L_1)$ or $\mathtt{inr}(L_1)$. Since we have shown that we reach stable terms, then $L_1$, $L_2$ are stable, so we can conclude that $L = (\mathtt{let}\ M\ \mathtt{be}\ P\ \mathtt{in}\ N)\bar{K}$ and thus $\mathtt{let}\ M\ \mathtt{be}\ P\ \mathtt{in}\ N$ are stable. □

**Lemma 5.16.** *If the term* $(\mathtt{let}\ (\mathtt{let}\ N\ \mathtt{be}\ P\ \mathtt{in}\ J)\ \mathtt{be}\ Q\ \mathtt{in}\ M)$ *is stable, then* $(\lambda PJ\ \mathtt{of}\ N\ \mathtt{is}\ Q\ \mathtt{in}\ M)$ *is stable.*

**Proof.** Let $\bar{K}$ be stable terms such that $L = (\lambda PJ\ \mathtt{of}\ N\ \mathtt{is}\ Q\ \mathtt{in}\ M)\bar{K}$ is not of functional type. If we show that $L$ is stable, then the lemma follows from Proposition 5.8.

For that, let us first show that the term $L$ is strongly normalizing. Consider any reduction sequence starting at $L$.

Since $(\mathtt{let}\ (\mathtt{let}\ N\ \mathtt{be}\ P\ \mathtt{in}\ J)\ \mathtt{be}\ Q\ \mathtt{in}\ M)$ is stable, then it is strongly normalizing by Lemma 5.11 and then $J$, $M$ and $N$ are strongly normalizing.

- If the outermost constructor $(\_\ \mathtt{of}\ \_\ \mathtt{is}\ \_\ \mathtt{in}\ \_)$ is never removed, then reductions proceed inside $M$, $N$, $J$ and $\bar{K}$.

  The terms $M$ and $N$ and $J$ are all strongly normalizing and the terms $\bar{K}$ are stable by hypothesis, so strongly normalizing by Lemma 5.11. That means that the reduction sequence necessarily terminates.
- If the outermost constructor $(\_\ \mathtt{of}\ \_\ \mathtt{is}\ \_\ \mathtt{in}\ \_)$ is removed, then by reasoning in the same way we did in Lemma 5.15, we know that the term $L$ reduces to some term having the form

  $$(\mathtt{let}\ (\mathtt{let}\ N'\ \mathtt{be}\ P\ \mathtt{in}\ J')\ \mathtt{be}\ Q\ \mathtt{in}\ M')\bar{K'}$$

  such that $N \longrightarrow^*_{TPC} N'$, $J \longrightarrow^*_{TPC} J'$, $M \longrightarrow^*_{TPC} M'$ and $\bar{K} \longrightarrow^*_{TPC} \bar{K'}$.

  By hypothesis the term $(\mathtt{let}\ (\mathtt{let}\ N\ \mathtt{be}\ P\ \mathtt{in}\ J)\ \mathtt{be}\ Q\ \mathtt{in}\ M)$ is stable, and thus by Proposition 5.8 the term

  $$(\mathtt{let}\ (\mathtt{let}\ N\ \mathtt{be}\ P\ \mathtt{in}\ J)\ \mathtt{be}\ Q\ \mathtt{in}\ M)\bar{K}$$

  is also stable. By Lemma 5.13 the term $(\mathtt{let}\ (\mathtt{let}\ N'\ \mathtt{be}\ P\ \mathtt{in}\ J')\ \mathtt{be}\ Q\ \mathtt{in}\ M')\bar{K'}$ turns out to be stable, and by Lemma 5.11 it turns to be strongly normalizing.

To finish the proof suppose the term $L$ reduces to a pair $\langle L_1, L_2\rangle$ or to $\mathtt{inl}(L_1)$ or to $\mathtt{inr}(L_1)$. Then we have necessarily removed the outermost constructors $(\_\ \mathtt{of}\ \_\ \mathtt{is}\ \_\ \mathtt{in}\ \_)$ and we obtain a reduction sequence from the term $(\mathtt{let}\ (\mathtt{let}\ N'\ \mathtt{be}\ P\ \mathtt{in}\ J')\ \mathtt{be}\ Q\ \mathtt{in}\ M')\bar{K'}$ to the term $\langle L_1, L_2\rangle$, $\mathtt{inl}(L_1)$ or $\mathtt{inr}(L_1)$. By definition of stability $L_1$, $L_2$ are stable, so we can conclude that both $L = (\lambda PJ\ \mathtt{of}\ N\ \mathtt{is}\ Q\ \mathtt{in}\ M)\bar{K}$ and $(\lambda PJ\ \mathtt{of}\ N\ \mathtt{is}\ Q\ \mathtt{in}\ M)$ are stable. □

We now establish the equivalence between stability of $(\lambda PJ)N$ and that of $\mathtt{let}\ N\ \mathtt{be}\ P\ \mathtt{in}\ J$.

**Lemma 5.17.** *If* $(\lambda P.J)N$ *is stable, then* $\texttt{let}\ N\ \texttt{be}\ P\ \texttt{in}\ J$ *is stable.*

**Proof.** Let $\bar{K}$ be stable terms such that $L = (\texttt{let}\ N\ \texttt{be}\ P\ \texttt{in}\ J)\bar{K}$ is not of functional type. If we show that $L$ is stable, then the lemma follows from Proposition 5.8.

For that, let us first show that $L$ is strongly normalizing. Consider any reduction sequence starting at $L$.

- If the outermost constructor *let* is never removed, then reductions proceed inside $N$, $J$ and $\bar{K}$. Since $(\lambda P.J)N$ is stable, then it is strongly normalizing by Lemma 5.11 and then $J$ and $N$ are strongly normalizing. The terms $\bar{K}$ are stable by hypothesis, so strongly normalizing by Lemma 5.11, so we can conclude that the reduction sequence necessarily terminates in this case.
- If the outermost constructor *let* is removed, then by reasoning in the same way we did in Lemma 5.15, we know that $L$ reduces to some term $T\bar{K'}$ such that $T \in [\![N,P,J]\!]$ and $\bar{K} \longrightarrow_{TPC} \bar{K'}$. Thus

$$\begin{array}{l}(\lambda P.J)N = \texttt{let}\ \lambda P.J\ \texttt{be}\ \sharp z\ \texttt{in}\ z\ \texttt{of}\ N\ \texttt{is}\ y\ \texttt{in}\ y \longrightarrow_{TPC} \\ \lambda P.J\ \texttt{of}\ N\ \texttt{is}\ y\ \texttt{in}\ y \longrightarrow^*_{TPC} \\ \texttt{let}\ (\texttt{let}\ N\ \texttt{be}\ P\ \texttt{in}\ J)\ \texttt{be}\ y\ \texttt{in}\ y \longrightarrow_{TPC} \\ \texttt{let}\ N\ \texttt{be}\ P\ \texttt{in}\ J \longrightarrow^*_{TPC} T\end{array}$$

  Since the term $(\lambda P.J)N$ is stable by hypothesis, $T$ turns out to be stable by Lemma 5.13. Now, the terms $\bar{K}$ are stable, so the terms $\bar{K'}$ are stable by Lemma 5.13 and $T\bar{K'}$ is stable by Proposition 5.8, thus strongly normalizing by Lemma 5.11. The reduction sequence terminates also in this case.

To finish the proof suppose the term $L$ reduces to a pair $\langle L_1, L_2\rangle$ or to $\texttt{inl}(L)$ or to $\texttt{inr}(L)$. Then we have necessarily removed the outermost constructor *let* we obtain a reduction sequence from the term $T\bar{K'}$ to the term $\langle L_1, L_2\rangle$, or $\texttt{inl}(L)$ or $\texttt{inr}(L)$. Since this term is stable, then $L_1$, $L_2$ are stable, so we can conclude that $L = (\texttt{let}\ N\ \texttt{be}\ P\ \texttt{in}\ J)\bar{K}$ and thus $\texttt{let}\ N\ \texttt{be}\ P\ \texttt{in}\ J$ are stable. □

**Lemma 5.18.** $\texttt{let}\ N\ \texttt{be}\ P\ \texttt{in}\ J$ *is stable, then* $(\lambda P.J)N$ *is stable.*

**Proof.** Let consider the term $(\lambda P.J)N$, an abbreviation of $\texttt{let}\ \lambda P.J\ \texttt{be}\ \sharp z\ \texttt{in}\ z\ \texttt{of}\ N\ \texttt{is}\ y\ \texttt{in}\ y$. By Lemma 5.15 it is sufficient to show that $[\![\lambda P.J, \#z, z\ \texttt{of}\ N\ \texttt{is}\ y\ \texttt{in}\ y]\!]$ is set-stable. We first remark that by hypothesis both $\lambda P.J$ and $z\ \texttt{of}\ N\ \texttt{is}\ y\ \texttt{in}\ y$ are strongly normalizing, so that we have to show that any term $T$ in the set is stable. Take any of such terms $T$. Then

$$\begin{array}{ll}\texttt{let}\ \lambda P.J\ \texttt{be}\ \sharp z\ \texttt{in}\ z\ \texttt{of}\ N\ \texttt{is}\ y\ \texttt{in}\ y & \longrightarrow^*_{TPC} \\ \texttt{let}\ \lambda P.J'\ \texttt{be}\ \sharp z\ \texttt{in}\ z\ \texttt{of}\ N'\ \texttt{is}\ y\ \texttt{in}\ y & \longrightarrow_{TPC} \\ \lambda P.J'\ \texttt{of}\ N'\ \texttt{is}\ y\ \texttt{in}\ y & \longrightarrow^*_{TPC} T\end{array}$$

where $J \longrightarrow^*_{TPC} J'$ and $N \longrightarrow^*_{TPC} N'$.

To show that $T$ is stable it is sufficient to show that $\lambda P.J'\ \texttt{of}\ N'\ \texttt{is}\ y\ \texttt{in}\ y$ is stable and by Lemma 5.16 it is sufficient to show the stability of the term $\texttt{let}\ (\texttt{let}\ N'\ \texttt{be}\ P\ \texttt{in}\ J')\ \texttt{be}\ y\ \texttt{in}\ y$. For that, it is sufficient to show, by Lemma 5.15,

that $[\![\texttt{let } N' \texttt{ be } P \texttt{ in } J', y, y]\!]$ is set-stable; and by definition of set-stability we need to show that $\texttt{let } N' \texttt{ be } P \texttt{ in } J'$ and $y$ are strongly normalizing, which is true by hypothesis, and that any term $H$ in the set is stable. Take one of such terms $H$. Then we have

$$\texttt{let (let } N' \texttt{ be } P \texttt{ in } J') \texttt{ be } y \texttt{ in } y \longrightarrow^*_{TPC} \texttt{let } R \texttt{ be } y \texttt{ in } y \longrightarrow_{TPC} R \longrightarrow^*_{TPC} H,$$

where $\texttt{let } N' \texttt{ be } P \texttt{ in } J' \longrightarrow^*_{TPC} R$. Now, since $\texttt{let } N \texttt{ be } P \texttt{ in } J$ is stable by hypothesis, then $\texttt{let } N' \texttt{ be } P \texttt{ in } J'$, $R$ and $H$ are stable by Lemma 5.13, so we can finally conclude that $(\lambda P.J)N$ is stable. □

The following notion is used in the proof of Lemma 5.20 to show that every typable term is stable. This measure takes into account the complexity of both patterns and terms, and this is in particular important when considering the cases of reduction rules like (*Cont*) or (*Pair*).

**Definition 5.4.** The measure of a term is a function $\mathscr{M}() : \mathscr{T}_{TPC} \to \mathbb{N}_2$, where $\mathbb{N}_2$ is the set of integers greater or equal than 2, defined as follows:

$$\begin{aligned}
\mathscr{M}(x) &= 2,\\
\mathscr{M}(\texttt{inl}(M)) &= \mathscr{M}(M)+1,\\
\mathscr{M}(\texttt{inr}(M)) &= \mathscr{M}(M)+1,\\
\mathscr{M}(\langle M,N\rangle) &= \mathscr{M}(M)+\mathscr{M}(N)+1,\\
\mathscr{M}(\lambda P.J) &= \mathscr{P}(P)+\mathscr{M}(J)+3,\\
\mathscr{M}(\texttt{let } M \texttt{ be } P \texttt{ in } N) &= \mathscr{P}(P)+\mathscr{M}(M)+\mathscr{M}(N),\\
\mathscr{M}([M \mid_\xi N]) &= \mathscr{M}(M)+\mathscr{M}(N)+1,\\
\mathscr{M}(T \texttt{ of } N \texttt{ is } Q \texttt{ in } L) &= \mathscr{M}(T)*\mathscr{M}(N)*\mathscr{P}(Q)+\mathscr{M}(L),
\end{aligned}$$

where the notation $\mathscr{P}(P)$ indicates the measure of the pattern $P$, defined as: $\mathscr{P}(x)=\mathscr{P}(\sharp z)=\mathscr{P}(\_)=2$ and $\mathscr{P}(@(P,Q))=\mathscr{P}(\langle P,Q\rangle)=\mathscr{P}((P \mid_\xi Q))=\mathscr{P}(P)+\mathscr{P}(Q)+3$.

Note that for abstractions, *let* terms and *of* terms, their measure $\mathscr{M}$ depends also on a measure $\mathscr{P}$ on patterns.

**Remark 5.19.** $\mathscr{M}(M) \geqslant \mathscr{M}(M\{\xi \leftarrow \mathtt{L}\}), \mathscr{M}(M\{\xi \leftarrow \mathtt{R}\})$.

**Lemma 5.20.** *Every typed term is stable.*

**Proof.** To prove this property we need a stronger property: Let $M$ be a term such that all its free *usual*[1] variables are among $\{x_i\}_{i=1\ldots n}$. If $N_1 \ldots N_n$ are stable terms and $\theta = \{x_1/N_1, \ldots x_n/N_n\}$ is a well-typed substitution, then $M\theta$ is stable.

The theorem follows by taking $N_i = x_i$ since variables are stable by Corollary 5.12. The proof proceeds by induction on $\mathscr{M}(M)$. In some of the cases that follow, in

[1] Hence, sum variables are not considered, here.

order to be able to apply the induction hypothesis to some sub-term $R$ of $M$ having a set of free usual variables $\{y_1,\dots y_m,x_1,\dots x_n\}$, we will make use of the substitution $\{x_1/N_1,\dots x_n/N_n,\overline{y}/\overline{y}\}$ (denoted by $\delta$), which verifies the hypothesis on substitutions because variables are stable by Corollary 5.12. Indeed, the term $R\delta = R\theta$ will be stable by induction hypothesis.

- $M \equiv x_i$. Then $x_i\theta = N_i$ and $N_i$ is stable by hypothesis.
- If $M \equiv \langle M_1, M_2\rangle$ we apply the induction hypothesis and Lemma 5.14.
- If $M \equiv \mathtt{inl}(N)$ or $M \equiv \mathtt{inr}(L)$ we apply the induction hypothesis and Lemma 5.14.
- If $M \equiv [M_1 \mid_\xi M_2]$, then we apply then induction hypothesis and Lemma 5.14.
- $M \equiv \mathtt{let}\ L\ \mathtt{be}\ P\ \mathtt{in}\ R$. We will show that the set $[\![L\theta, P, R\theta]\!]$ is set-stable, so that the term $M\theta = \mathtt{let}\ L\theta\ \mathtt{be}\ P\ \mathtt{in}\ R\theta$ will be stable by Lemma 5.15. That is, we will show:
  1. Every term $T$ in $\mathscr{S} = [\![L\theta, P, R\theta]\!]$ is stable.
  2. $L\theta$ and $R\theta$ are strongly normalizing.

  Now, let $\{y_1,\dots y_m\}(m \geqslant 0)$ be the set of usual variables of the pattern $P$. By $\alpha$-conversion we can assume that $\{y_1,\dots y_m\} \cap \{x_1,\dots x_n\} = \emptyset$ and also that $\{y_1,\dots y_m\} \cap \bigcup_{i=1\dots n} FV(N_i) = \emptyset$. Thus, by the induction hypothesis, $L\theta$ and $R\delta = R\theta$ are stable for $\delta = \{x_1/N_1,\dots x_n/N_n, \overline{y}/\overline{y}\}$ (since variables are stable by Corollary 5.12), thus they are strongly normalizing by Lemma 5.11).

Now, to show that every term in $\mathscr{S}$ is stable we proceed by cases:

– $P = \_$. If $T \in \mathscr{S}$, then we have necessarily a reduction sequence having the form:

$$\begin{array}{ll} \mathtt{let}\ L\theta\ \mathtt{be}\ \_\ \mathtt{in}\ R\theta & \longrightarrow^*_{TPC} \\ \mathtt{let}\ L'\ \mathtt{be}\ \_\ \mathtt{in}\ R' & \longrightarrow_{TPC} \\ R' & \longrightarrow^*_{TPC}\ T \end{array}$$

where $L\theta \longrightarrow^*_{TPC} L'$ and $R\theta \longrightarrow^*_{TPC} R'$. Since $R\theta$ is stable, then $T$ is stable by Lemma 5.13.

– $P = x$. If $T \in \mathscr{S}$, then we have necessarily a reduction sequence having the form:

$$\begin{array}{ll} \mathtt{let}\ L\theta\ \mathtt{be}\ x\ \mathtt{in}\ R\theta & \longrightarrow^*_{TPC} \\ \mathtt{let}\ L'\ \mathtt{be}\ x\ \mathtt{in}\ R' & \longrightarrow_{TPC} \\ R'\{x \leftarrow L'\} & \longrightarrow^*_{TPC}\ T \end{array}$$

where $L\theta \longrightarrow^*_{TPC} L'$ and $R\theta \longrightarrow^*_{TPC} R'$. By $\alpha$-conversion we can suppose that $(R\theta)\{x \leftarrow L'\} = R(\theta \cup \{x \leftarrow L'\})$. We know that $L\theta$ is stable by the induction hypothesis. Thus, since $L\theta$ reduces to $L'$, $L'$ is stable by Lemma 5.13.

Our aim is to show that the term $R'\{x \leftarrow L'\}$ is stable, since, given that it reduces to $T$, its stability implies the stability of $T$, by Lemma 5.13.

Now, the term $R(\theta \cup \{x \leftarrow L'\})$ is stable by the induction hypothesis (since $L'$ is stable). Moreover, since $R\theta$ reduces to $R'$, we have by Lemma 5.2 that $(R\theta)\{x \leftarrow L'\} \longrightarrow^*_{TPC} R'\{x \leftarrow L'\}$. Hence, Lemma 5.13 allows us to conclude that $R'\{x \leftarrow L'\}$ is stable.

– $P = \sharp z$. If $T \in \mathscr{S}$, then we have necessarily a reduction sequence having the form:

$$\begin{array}{ll} \texttt{let } L\theta \texttt{ be } \sharp z \texttt{ in } R\theta & \longrightarrow^*_{TPC} \\ \texttt{let } \lambda Q.J \texttt{ be } \sharp z \texttt{ in } R' & \longrightarrow_{TPC} \\ R'\{z \leftarrow \lambda Q.J\} & \longrightarrow^*_{TPC} T \end{array}$$

where $L\theta \longrightarrow^*_{TPC} \lambda Q.J$ and $R\theta \longrightarrow^*_{TPC} R'$. By α-conversion we may suppose that $(R\theta)\{z \leftarrow \lambda Q.J\} = R(\theta \cup \{z \leftarrow \lambda Q.J\})$ and since $L\theta$ is stable by hypothesis, then $\lambda Q.J$ is stable by Lemma 5.13, hence $R(\theta \cup \{z \leftarrow \lambda Q.J\})$ is stable by induction hypothesis. Now, by Lemma 5.2 we have that $(R\theta)\{z \leftarrow \lambda Q.J\} \longrightarrow^*_{TPC} R'\{z \leftarrow \lambda Q.J\}$ and thus $R'\{z \leftarrow \lambda Q.J\}$ and $T$ are stable by Lemma 5.13.

– $P = @(P_1, P_2)$. If $T \in \mathscr{S}$, then we have necessarily a reduction sequence having the form:

$$\begin{array}{ll} \texttt{let } L\theta \texttt{ be } @(P_1,P_2) \texttt{ in } R\theta & \longrightarrow^*_{TPC} \\ \texttt{let } L' \texttt{ be } @(P_1,P_2) \texttt{ in } R' & \longrightarrow_{TPC} \\ \texttt{let } L' \texttt{ be } P_1 \texttt{ in } (\texttt{let } L' \texttt{ be } P_2 \texttt{ in } R') & \longrightarrow^*_{TPC} T \end{array}$$

where $L\theta \longrightarrow^*_{TPC} L'$ and $R\theta \longrightarrow^*_{TPC} R'$.

We can write the term

$$\texttt{let } L' \texttt{ be } P_1 \texttt{ in } (\texttt{let } L' \texttt{ be } P_2 \texttt{ in } R\theta)$$

in another way, namely as

$$(\texttt{let } w \texttt{ be } P_1 \texttt{ in } (\texttt{let } w \texttt{ be } P_2 \texttt{ in } R))(\theta \cup \{w \leftarrow L'\}),$$

where $w$ is a fresh variable.

Since $L'$ is stable by Lemma 5.13, then the substitution $\theta \cup \{w \leftarrow L'\}$ verifies the hypothesis of this lemma.

Now,

$$\begin{aligned} &\mathscr{M}(\texttt{let } w \texttt{ be } P_1 \texttt{ in } (\texttt{let } w \texttt{ be } P_2 \texttt{ in } R)) \\ &\quad = 2 + 2 + \mathscr{P}(P_1) + \mathscr{P}(P_2) + \mathscr{M}(R) \\ &\quad < 2 + \mathscr{P}(P_1) + \mathscr{P}(P_2) + 3 + \mathscr{M}(R) \\ &\quad = \mathscr{M}(\texttt{let } w \texttt{ be } @(P_1,P_2) \texttt{ in } R). \end{aligned}$$

On the other hand, $2 = \mathscr{M}(w) \leqslant \mathscr{M}(L)$ implies

$$\mathscr{M}(\texttt{let } w \texttt{ be } @(P_1,P_2) \texttt{ in } R) \leqslant \mathscr{M}(\texttt{let } L \texttt{ be } @(P_1,P_2) \texttt{ in } R)$$

hence $\mathscr{M}(\texttt{let } w \texttt{ be } P_1 \texttt{ in } (\texttt{let } w \texttt{ be } P_2 \texttt{ in } R)) < \mathscr{M}(\texttt{let } L \texttt{ be } @(P_1,P_2) \texttt{ in } R)$.

Thus, we can apply the i.h. to the term $\texttt{let } w \texttt{ be } P_1 \texttt{ in } (\texttt{let } w \texttt{ be } P_2 \texttt{ in } R)$ and infer that $(\texttt{let } w \texttt{ be } P_1 \texttt{ in } (\texttt{let } w \texttt{ be } P_2 \texttt{ in } R))(\theta \cup \{w \leftarrow L'\})$ is stable. Now, this last term is equal to $\texttt{let } L' \texttt{ be } P_1 \texttt{ in } (\texttt{let } L' \texttt{ be } P_2 \texttt{ in } R\theta)$, and we have the

reduction sequence

$$\begin{array}{l}\texttt{let } L' \texttt{ be } P_1 \texttt{ in } (\texttt{let } L' \texttt{ be } P_2 \texttt{ in } R\theta) \longrightarrow^*_{TPC} \\ \texttt{let } L' \texttt{ be } P_1 \texttt{ in } (\texttt{let } L' \texttt{ be } P_2 \texttt{ in } R') \longrightarrow^*_{TPC} T\end{array}$$

Hence, by Lemma 5.13, we can finally conclude that $T$ is stable.

– $P = (P_1 \mid_\xi P_2)$. If $T \in \mathscr{S}$, then without loss of generality we can suppose that we have a reduction sequence having the form:

$$\begin{array}{ll}\texttt{let } L\theta \texttt{ be } (P_1 \mid_\xi P_2) \texttt{ in } R\theta & \longrightarrow^*_{TPC} \\ \texttt{let inl}(L') \texttt{ be } (P_1 \mid_\xi P_2) \texttt{ in } R' & \longrightarrow_{TPC} \\ \texttt{let } L' \texttt{ be } P_1 \texttt{ in } R'\{\xi \leftarrow \texttt{L}\} & \longrightarrow^*_{TPC} T\end{array}$$

By α-conversion we may suppose that there is no capture of variables w.r.t the substitution $\theta$ so that $R\theta\{\xi \leftarrow \texttt{L}\} = R\{\xi \leftarrow \texttt{L}\}\theta$ by the substitution lemma [5]. Also, since $L\theta$ is stable by induction hypothesis, then $\texttt{inl}(L')$ is stable by Lemma 5.13 and $L'$ is stable by definition. Now, the term $\texttt{let } L' \texttt{ be } P_1 \texttt{ in } R\theta\{\xi \leftarrow \texttt{L}\}$ can be written as

$$(\texttt{let } w \texttt{ be } P_1 \texttt{ in } R\{\xi \leftarrow \texttt{L}\})(\theta \cup \{w \leftarrow L'\}),$$

where $w$ is a fresh variable and the substitution $\theta \cup \{w \leftarrow L'\}$ verifies the hypothesis of the lemma. Since $\mathscr{M}(\texttt{let } w \texttt{ be } P_1 \texttt{ in } R\{\xi \leftarrow \texttt{L}\}) = 2 + \mathscr{M}(P_1) + \mathscr{M}(R\{\xi \leftarrow \texttt{L}\})$ which is strictly smaller than $\mathscr{M}(L) + \mathscr{M}(P_1) + \mathscr{M}(P_2) + 3 + \mathscr{M}(R) = \mathscr{M}(\texttt{let } L \texttt{ be } (P_1 \mid_\xi P_2) \texttt{ in } R)$, then we conclude that $(\texttt{let } w \texttt{ be } P_1 \texttt{ in } R\{\xi \leftarrow \texttt{L}\})(\theta \cup \{w \leftarrow L'\})$ is stable by induction hypothesis.
Now,

$$\begin{array}{l}(\texttt{let } w \texttt{ be } P_1 \texttt{ in } R\{\xi \leftarrow \texttt{L}\})(\theta \cup \{w \leftarrow L'\}) \\ \quad = \texttt{let } L' \texttt{ be } P_1 \texttt{ in } R\theta\{\xi \leftarrow \texttt{L}\} \\ \quad \longrightarrow^*_{TPC} \texttt{let } L' \texttt{ be } P_1 \texttt{ in } R'\{\xi \leftarrow \texttt{L}\}\end{array}$$

(by Lemma 5.2). Thus, by Lemma 5.13, we are done.

– $P = \langle P_1, P_2 \rangle$. If $T \in \mathscr{S}$, then we have necessarily a reduction sequence having the form:

$$\begin{array}{ll}\texttt{let } L\theta \texttt{ be } \langle P_1, P_2 \rangle \texttt{ in } R\theta & \longrightarrow^*_{TPC} \\ \texttt{let } \langle M_1, M_2 \rangle \texttt{ be } \langle P_1, P_2 \rangle \texttt{ in } R' & \longrightarrow_{TPC} \\ \texttt{let } M_1 \texttt{ be } P_1 \texttt{ in } (\texttt{let } M_2 \texttt{ be } P_2 \texttt{ in } R') & \longrightarrow^*_{TPC} T\end{array}$$

where $L\theta \longrightarrow^*_{TPC} \langle M_1, M_2 \rangle$ and $R\theta \longrightarrow^*_{TPC} R'$. Since $L\theta$ is stable by the induction hypothesis, then $\langle M_1, M_2 \rangle$ is stable by Lemma 5.13 and $M_1, M_2$ are stable by definition.
We can then write the term

$$\texttt{let } M_1 \texttt{ be } P_1 \texttt{ in } (\texttt{let } M_2 \texttt{ be } P_2 \texttt{ in } R\theta)$$

as

$$(\texttt{let } w_1 \texttt{ be } P_1 \texttt{ in } (\texttt{let } w_2 \texttt{ be } P_2 \texttt{ in } R))(\theta \cup \{w_1/M_1, w_2/M_2\}),$$

where $w_1$ and $w_2$ are fresh variables. Since

$$\begin{aligned}&\mathscr{M}(\texttt{let } w_1 \texttt{ be } P_1 \texttt{ in } (\texttt{let } w_2 \texttt{ be } P_2 \texttt{ in } R))\\ &= 4 + \mathscr{P}(P_1) + \mathscr{P}(P_2) + \mathscr{M}(R)\\ &< \mathscr{M}(L) + \mathscr{P}(P_1) + \mathscr{P}(P_2) + 3 + \mathscr{M}(R)\\ &= \mathscr{M}(\texttt{let } L \texttt{ be } \langle P_1, P_2\rangle \texttt{ in } R)\end{aligned}$$

then the term $\texttt{let } M_1 \texttt{ be } P_1 \texttt{ in } (\texttt{let } M_2 \texttt{ be } P_2 \texttt{ in } R\theta)$ is stable by induction hypothesis and thus $\texttt{let } M_1 \texttt{ be } P_1 \texttt{ in } (\texttt{let } M_2 \texttt{ be } P_2 \texttt{ in } R')$ and $T$ are also stable by Lemma 5.13.

- $M \equiv L \texttt{ of } U \texttt{ is } Q \texttt{ in } R$.
  Let $Var(Q) = \{y_1, \ldots, y_m\}$ $(m \geqslant 0)$. By α-conversion we can assume that $\{y_1, \ldots, y_m\} \cap \{x_1, \ldots, x_n\} = \emptyset$ and also $\{y_1, \ldots, y_m\} \cap \bigcup_{i=1\ldots n} FV(N_i) = \emptyset$ so by i.h. $U\theta$ and $R\delta = R\theta$ for $\delta = \{\theta, \overline{y}/\overline{y}\}$ are stable since variables are stable by Corollary 5.12.
  If $L\theta$ is a variable, then the property holds by Lemma 5.10. Otherwise, $L\theta$ is a $\lambda$-abstraction $\lambda P.J$ (that is stable by i.h.) and we proceed as follows. First, we remark the following facts:
  1. The term $\texttt{let } U\theta \texttt{ be } P \texttt{ in } J$ is stable. In fact, since $\lambda P.J$ is stable, $(\lambda P.J)U\theta$ is stable by definition and $\texttt{let } U\theta \texttt{ be } P \texttt{ in } J$ is stable by Lemma 5.17.
  2. $\mathscr{M}(\texttt{let } z \texttt{ be } Q \texttt{ in } R) = 2 + \mathscr{P}(Q) + \mathscr{M}(R) < \mathscr{M}(L) * \mathscr{M}(U) * \mathscr{P}(Q) + \mathscr{M}(R) = \mathscr{M}(M)$, where $z$ is a fresh variable.

  Now, to show that $\lambda P.J \texttt{ of } U\theta \texttt{ is } Q \texttt{ in } R\theta$ is stable it is sufficient to show, by Lemma 5.16, that $M' = \texttt{let } (\texttt{let } U\theta \texttt{ be } P \texttt{ in } J) \texttt{ be } Q \texttt{ in } R\theta$ is stable. Since $M'$ can be written as

  $$(\texttt{let } z \texttt{ be } Q \texttt{ in } R)(\theta \cup \{z \leftarrow \texttt{let } U\theta \texttt{ be } P \texttt{ in } J\}),$$

  where $\mathscr{M}(\texttt{let } z \texttt{ be } Q \texttt{ in } R) < \mathscr{M}(M)$ and the term $\texttt{let } U\theta \texttt{ be } P \texttt{ in } J$ is stable, then $M'$ turns out to be stable by i.h. and therefore $M\theta$ is stable.
- $M \equiv \lambda P.J$ and $(\lambda P.J)\theta \equiv \lambda P.J\theta$.
  Consider any stable term $R$ of appropriate type. If $(\lambda P.J\theta)R$ is shown to be stable, $(\lambda P.J\theta)$ is stable by definition of stability. By Lemma 5.18 it is sufficient to show that $\texttt{let } R \texttt{ be } P \texttt{ in } J\theta$ is stable. Since $\texttt{let } R \texttt{ be } P \texttt{ in } J\theta$ can be written as

  $$(\texttt{let } w \texttt{ be } P \texttt{ in } J)(\theta \cup \{w \leftarrow R\}),$$

  where $R$ is stable by hypothesis, and

  $$\mathscr{M}(\texttt{let } w \texttt{ be } P \texttt{ in } J) = 2 + \mathscr{P}(P) + \mathscr{M}(J) < \mathscr{P}(P) + \mathscr{M}(J) + 3 = \mathscr{M}(\lambda P.J)$$

  we can conclude by induction hypothesis that $\texttt{let } R \texttt{ be } P \texttt{ in } J\theta$ is stable. □

By Lemmas 5.20 and 5.11 we obtain:

**Theorem 5.21** (Strong normalization for *TPC*). *If* $\Gamma \rhd_{TPC} M : A$ *then* $M$ *is TPC-strongly normalizing.*

## 6. Fundamental properties of $TPC_{\mathrm{ES}}$

This section establishes important properties for $TPC_{\mathrm{ES}}$, namely: confluence, preservation of $TPC$-strong normalization, strong normalization for well-typed terms, and reducibility to "values" for closed well-typed terms.

As regards the proof technique used to show confluence we use the interpretation method [21], which *maps* reduction sequences of a rewriting relation to sequences of another relation for which we already know confluence. This is the standard technique used to show confluence of calculi with explicit substitutions. In our case, $TPC_{\mathrm{ES}}$-reduction is simulated by $TPC$-reduction, so that confluence of $TPC_{\mathrm{ES}}$ is a consequence of confluence of $TPC$. Note that $TPC_{\mathrm{ES}}$ is not "orthogonal" so that we can not directly apply the Tait and Martin-Löf's [5] technique for $TPC_{\mathrm{ES}}$ as done for $TPC$.

To prove strong normalization for $TPC_{\mathrm{ES}}$ we also use an intermediate result stating strong normalization for $TPC$. This is also a standard approach to establish strong normalization for calculi with explicit substitutions as no direct and general techniques are known to deal with them.

**Lemma 6.1.** *$ES$-normal forms of $TPC_{\mathrm{ES}}$-terms are pure terms.*

**Proof.** By induction on the structure of $TPC_{\mathrm{ES}}$-terms using Lemma 2.7. □

**Lemma 6.2.**
- *If $M \longrightarrow_{TPC_{\mathrm{ES}}} N$, then $ES(M) \longrightarrow^*_{TPC} ES(N)$.*
- *If $M \longrightarrow_{TPC_{\mathrm{ES}}} N$ is a head $R$-reduction, where $R = TPC_{\mathrm{ES}} \setminus ES$, then $ES(M) \longrightarrow^+_{TPC} ES(N)$.*

**Proof.** By induction on the structure of $TPC_{\mathrm{ES}}$-terms using Lemma 2.7. □

**Lemma 6.3.** *If $M \longrightarrow_{TPC} N$, then $M \longrightarrow^+_{TPC_{\mathrm{ES}}} N$.*

**Proof.** By induction on the structure of $TPC_{\mathrm{ES}}$-terms using Lemma 2.7. □

**Corollary 6.4.** *If $M \in SN_{TPC_{\mathrm{ES}}}$, then $ES(M) \in SN_{TPC}$.*

**Proof.** By *reductio ad absurdum*, using Lemma 6.3 and the fact that $M \longrightarrow^*_{TPC_{\mathrm{ES}}} ES(M)$. □

Thus, we get the following result:

**Theorem 6.5** (Confluence of $TPC_{\mathrm{ES}}$). *The relation $TPC_{\mathrm{ES}}$ is confluent.*

**Proof.** By the interpretation method [21], using the fact that $TPC$ is confluent (by Theorem 5.7) and Lemmas 6.3 and 6.2. □

In contrast to strong normalization, which is a property usually proved for *well-typed* terms, preservation of strong normalization is a property on *raw* terms, and it

is, in some sense, a property that guarantees the correctness of the reduction relation with explicit substitutions with respect to the corresponding reduction relation with meta-level substitutions.

**Definition 6.1** ($S$ preserves $R$-strong normalization). Let $(R, \mathscr{T}_R)$ and $(S, \mathscr{T}_S)$ be two reduction systems such that $\mathscr{T}_R \subseteq \mathscr{T}_S$. We say that $S$ preserves $R$-strong normalization (or that PSN for $S$ holds) if every term in $\mathscr{T}_R$ which is $R$-strongly normalizing is also $S$-strongly normalizing.

To show that $\longrightarrow_{TPC_{\mathrm{ES}}}$ preserves $\longrightarrow_{TPC}$ -strong normalization of raw $TPC$-terms, we introduce the following definition, where the notation $SN_R$ is used to denote the set of all terms that are $R$-strongly normalizing and the notation $N \subseteq M$ is used as a shorthand for "$N$ is a sub-term of $M$".

**Definition 6.2.** The set $\mathscr{F}$ is defined as $\{M \mid M \in \mathscr{T}_{TPC_{\mathrm{ES}}} \text{ and } \forall N \subseteq M \ ES(N) \in SN_{TPC}\}$.

The set $\mathscr{F}$ is not only stable by $TPC_{\mathrm{ES}}$-reduction (Lemma 6.6) but also contains only terms which are $TPC_{\mathrm{ES}}$-strongly normalizable (Lemma 6.10).

**Lemma 6.6.** *$\mathscr{F}$ is closed under $TPC_{\mathrm{ES}}$ reduction.*

**Proof.** By induction on the structure of $M$ using the definition of $\mathscr{F}$ and Lemma 6.2. □

In order to show that $\longrightarrow_{TPC_{\mathrm{ES}}}$ preserves $\longrightarrow_{TPC}$ -strong normalization, we define the signature $\Sigma$ as

$$\{App_n, [\,]_n, Cut_n \mid n \geqslant 0\} \cup \{\circledast, Inl, Inr, Case, Pair, Lambda\},$$

where the respective arities of the function symbols are: $\circledast$ is a constant, $Inl$, $Inr$ and $Lambda$ are unary, $Cut_n$, $Pair$,$Case$, $[\,]_n$ are binary and $App_n$ is ternary. We define a precedence on $\Sigma$ by stating that for all $n \geqslant 0$:

$$App_{n+1} \succ [\,]_n \succ Cut_n \succ App_n \quad \text{and} \quad Cut_n \succ \circledast, Inl, Inr, Case, Pair, Lambda.$$

We assume that all symbols have multi-set status. As the precedence $\succ$ is well-founded on symbols, then $>_{rpo}$ is well-founded (see for example [14,26]) on the set of all the terms over the signature $\Sigma$.

We denote by $maxred_{TPC}(M)$ the maximal length of a $TPC$-reduction sequence starting at $M$.

We define the translation $\mathscr{R}(L)$ from $\mathscr{F}$ to $\mathscr{T}_\Sigma$ by induction on the structure of $L$ as follows:

$$\begin{aligned}
\mathscr{R}(x) &= \circledast,\\
\mathscr{R}(\langle M,N\rangle) &= Pair(\mathscr{R}(M),\mathscr{R}(N)),\\
\mathscr{R}(\lambda P{:}A.M) &= Lambda(\mathscr{R}(M)),\\
\mathscr{R}(\mathtt{inl}(M)) &= Inl(\mathscr{R}(M)),\\
\mathscr{R}(\mathtt{inr}(M)) &= Inr(\mathscr{R}(M)),\\
\mathscr{R}([M\mid_\xi N]) &= Case(\mathscr{R}(M),\mathscr{R}(N)),\\
\mathscr{R}(T\ \mathtt{of}\ M\ \mathtt{is}\ P{:}A\ \mathtt{in}\ N) &= App_n(\mathscr{R}(T),\mathscr{R}(M),\mathscr{R}(N)),\\
\mathscr{R}(\mathtt{let}\ M\ \mathtt{be}\ P{:}A\ \mathtt{in}\ N) &= Cut_n(\mathscr{R}(M),\mathscr{R}(N)),\\
\mathscr{R}(M[x/N]) &= \mathscr{R}(M)[\mathscr{R}(N)]_n,
\end{aligned}$$

where the label $n$ appearing on the right-hand side of the last three lines of the definition is $maxred_{TPC}(ES(L))$.

**Remark 6.7.** $\mathscr{R}(M) \geq_{rpo} \mathscr{R}(M\{\xi \leftarrow \mathtt{K}\})$.

**Remark 6.8.** If $M$ is in $\mathscr{F}$, then $\mathscr{R}(M)$ is well-defined since $ES(N)$ is in $SN_{TPC}$ for every sub-term $N$ of $M$.

Next the lemma shows that $TPC_{\mathrm{ES}}$-reduction is decreasing with respect to the order $>_{rpo}$. This will allow us to first prove that $\mathscr{F}$ only contains terms which are $TPC_{\mathrm{ES}}$-strongly normalizable, as already announced, and as a consequence, to obtain the preservation of $TPC$-strong normalization.

**Lemma 6.9.** *For every* $M \in \mathscr{F}$, *if* $M \longrightarrow_{TPC_{\mathrm{ES}}} N$, *then* $\mathscr{R}(M) >_{rpo} \mathscr{R}(N)$.

**Proof.** We remark that for the base case $M \equiv x$ the property vacuously holds. Also, we know by Lemma 6.2 that if $M \longrightarrow_{TPC_{\mathrm{ES}}} N$ is a head $R$-reduction with $R = TPC_{\mathrm{ES}} \setminus ES$, then if $\mathscr{R}(M)$ has an external label, then it must be strictly greater to that of $\mathscr{R}(N)$ (if it has one). Also, the label of a sub-term $S$ of a given term $T$ is smaller or equal than the label of $T$ itself. We now show all the other cases:

- If $M = \lambda Q : B.M_1\ \mathtt{of}\ M_2\ \mathtt{is}\ P : A\ \mathtt{in}\ M_3 \rightarrow_{of} \mathtt{let}\ (\mathtt{let}\ M_2\ \mathtt{be}\ Q : B\ \mathtt{in}\ M_1)\ \mathtt{be}\ P : A\ \mathtt{in}\ M_3 = N$, then

$$\mathscr{R}(M) = App_m(Lambda(\mathscr{R}(M_1)),\mathscr{R}(M_2),\mathscr{R}(M_3)),$$
$$\text{where } m = maxred_{TPC}(ES(M)) \text{ and}$$
$$\mathscr{R}(N) = Cut_n((Cut_k(\mathscr{R}(M_2),\mathscr{R}(M_1))),\mathscr{R}(M_3)),$$

  where $n = maxred_{TPC}(ES(N))$ and $k = maxred_{TPC}(ES(\mathtt{let}\ M_2\ \mathtt{be}\ Q : B\ \mathtt{in}\ M_1))$. Since $m > n \geqslant k$, then $App_m \succ Cut_n$ and $App_m \succ Cut_k$ so we are done by the definition of the precedence.

- If $M = \texttt{let}\ \langle M_1, M_2\rangle\ \texttt{be}\ \langle P_1, P_2\rangle\ \texttt{in}\ L \longrightarrow_{Pair} \texttt{let}\, M_1\, \texttt{be}\, P_1\, \texttt{in}\ (\texttt{let}\, M_2\, \texttt{be}\, P_2\ \texttt{in}\ L) = N$, then

$$\mathscr{R}(M) = Cut_m(Pair(\mathscr{R}(M_1), \mathscr{R}(M_2)), \mathscr{R}(L)), \quad \text{where}$$
$$m = maxred_{TPC}(ES(M)) \text{ and}$$
$$\mathscr{R}(N) = Cut_n(\mathscr{R}(M_1), Cut_k(\mathscr{R}(M_2), \mathscr{R}(L))),$$

  where $n = maxred_{TPC}(ES(N))$ and $k = maxred_{TPC}(ES(\texttt{let}\ M_2\ \texttt{be}\ P_2\ \texttt{in}\ L))$. Since $m > n \geqslant k$, then $Cut_m \succ Cut_n$ and $Cut_m \succ Cut_k$ and so we are done by the definition of the precedence.
- If $M = \texttt{let}\, R\, \texttt{be}\ @(P_1, P_2)\ \texttt{in}\ L \longrightarrow_{Cont} \texttt{let}\, R\, \texttt{be}\, P_1\ \texttt{in}\ (\texttt{let}\, M\ \texttt{be}\, P_2\ \texttt{in}\ L) = N$, then

$$\mathscr{R}(M) = Cut_m(\mathscr{R}(R), \mathscr{R}(L)), \quad \text{where } m = maxred_{TPC}(ES(M)) \text{ and}$$
$$\mathscr{R}(N) = Cut_n(\mathscr{R}(R), Cut_k(\mathscr{R}(R), \mathscr{R}(L))),$$

  where $n = maxred_{TPC}(ES(N))$ and $k = maxred_{TPC}(ES(\texttt{let}\ R\ \texttt{be}\ P_2\ \texttt{in}\ L))$. Since $m > n \geqslant k$, then $Cut_m \succ Cut_n$ and $Cut_m \succ Cut_k$ so we are done by definition of the precedence.
- If $M = \texttt{let}\ (\lambda P.J)\ \texttt{be}\ \sharp z\ \texttt{in}\ L \longrightarrow_{Lambda} L[z/\lambda P.J]$, then

$$\mathscr{R}(M) = Cut_m(\mathscr{R}(\lambda P.J), \mathscr{R}(L)),$$
$$\mathscr{R}(N) = \mathscr{R}(L)[\mathscr{R}(\lambda P.J)]_n,$$

  where $m = maxred_{TPC}(ES(M))$ and $n = maxred_{TPC}(ES(N))$. Since $m > n$, then $Cut_m \succ [\ ]_n$ and so we are done.
- If $M = \texttt{let}\ M_1\ \texttt{be}\ x\ \texttt{in}\ M_2 \longrightarrow_{Sub} M_2[x/M_1] = N$, then

$$\mathscr{R}(M) = Cut_m(\mathscr{R}(M_1), \mathscr{R}(M_2)),$$
$$\mathscr{R}(N) = \mathscr{R}(M_2)[\mathscr{R}(M_1)]_n,$$

  where $m = maxred_{TPC}(ES(M))$ and $n = maxred_{TPC}(ES(N))$. Since $m > n$, then $Cut_m \succ [\ ]_n$ so we are done by definition of the precedence.
- If $M = \texttt{let inl}(L)\ \texttt{be}\ (P\ |_\xi\ Q)\ \texttt{in}\ R \longrightarrow_{Case1} \texttt{let}\ L\ \texttt{be}\ P\ \texttt{in}\ R\{\xi \leftarrow \texttt{L}\} = N$, then

$$\mathscr{R}(M) = Cut_m(Inl(\mathscr{R}(L)), \mathscr{R}(R)),$$
$$\mathscr{R}(N) = Cut_n(\mathscr{R}(L), \mathscr{R}(R\{\xi \leftarrow \texttt{L}\})),$$

  where $m = maxred_{TPC}(ES(M))$ and $n = maxred_{TPC}(ES(N))$ and $m > n$. Since $Cut_m \succ Cut_n$, the property holds by definition of *RPO*.
- The cases $M \longrightarrow_{Var1,Var2,Weak} N$ are trivial by the sub-term property of $>_{rpo}$.

- If $M = \langle M_1, M_2\rangle[\theta] \longrightarrow_{Dpair} \langle M_1[\theta], M_2[\theta]\rangle = N$, then

$$\mathscr{R}(M) = Pair(\mathscr{R}(M_1), \mathscr{R}(M_2))[\mathscr{R}(L)]_m,$$

$$\mathscr{R}(N) = Pair(\mathscr{R}(M_1)[\mathscr{R}(L)]_{k_1}, \mathscr{R}(M_2)[\mathscr{R}(L)]_{k_2}),$$

  where $m = maxred_{TPC}(ES(M))$, $k_1 = maxred_{TPC}(ES(M_1[\theta]))$, $k_2 = maxred_{TPC}(ES(M_2[\theta]))$ and $m \geqslant k_1, k_2$. Since $[\ ]_m \succ Pair$ and $[\ ]_n$ has multi-set status the property holds by definition.
- The cases $M \longrightarrow_{Dlambda,Dinl,Dinr,Dcase} N$ are similar to the previous one.
- If $M = (\texttt{let}\ M_1\ \texttt{be}\ P\ \texttt{in}\ M_2)[\theta] \longrightarrow_{Dcut} \texttt{let}\ M_1[\theta]\ \texttt{be}\ P\ \texttt{in}\ M_2[\theta] = N$, then

$$\mathscr{R}(M) = Cut_p(\mathscr{R}(M_1), \mathscr{R}(M_2))[\mathscr{R}(\theta)]_n,$$

$$\mathscr{R}(N) = Cut_n(\mathscr{R}(M_1)[\mathscr{R}(\theta)]_{k_1}, \mathscr{R}(M_2)[\mathscr{R}(\theta)]_{k_2}),$$

  where $n = maxred_{TPC}(ES(M)) = maxred_{TPC}(ES(N))$, $k_1 = maxred_{TPC}(ES(M_1[\theta]))$, $k_2 = maxred_{TPC}(ES(M_2[\theta]))$ and $m > n \geqslant k_1, k_2$. Since $[\ ]_n \succ Cut_n$, then the property follows by definition.
- If $M = (T\ \texttt{of}\ M_1\ \texttt{is}\ P{:}A\ \texttt{in}\ M_2)[\theta] \longrightarrow_{Dapp} T[\theta]\ \texttt{of}\ M_1[\theta]\ \texttt{is}\ P{:}A\ \texttt{in}\ M_2[\theta] = N$, then

$$\mathscr{R}(M) = App_p(\mathscr{R}(T), \mathscr{R}(M_1), \mathscr{R}(M_2))[\mathscr{R}(\theta)]_n,$$

$$\mathscr{R}(N) = App_n(\mathscr{R}(T)[\mathscr{R}(\theta)]_t, \mathscr{R}(M_1)[\mathscr{R}(\theta)]_{k_1}, \mathscr{R}(M_2)[\mathscr{R}(\theta)]_{k_2}),$$

  where $n = maxred_{TPC}(ES(M)) = maxred_{TPC}(ES(N))$, $t = maxred_{TPC}(ES(T[\theta]))$, $k_1 = maxred_{TPC}(ES(M_1[\theta]))$, $k_2 = maxred_{TPC}(ES(M_2[\theta]))$ and $n \geqslant t, k_2, k_3$. Since $[\ ]_n \succ App_n$, then the property follows by definition.
- The context cases are simple. We just show here the case: $M = \texttt{let}\ M_1\ \texttt{be}\ P : A\ \texttt{in}\ M_2 \longrightarrow_{TPC_{ES}} \texttt{let}\ M_1'\ \texttt{let}\ P : A\ \texttt{in}\ M_2 = N$, where $M_1 \longrightarrow_{TPC_{ES}} M_1'$. By i.h. $\mathscr{R}(M_1) >_{rpo} \mathscr{R}(M_1')$ and obviously we have $m = maxred_{TPC}(ES(M)) \geqslant maxred_{TPC}(ES(N)) = n$ so that $\mathscr{R}(M) = Cut_m(\mathscr{R}(M_1), \mathscr{R}(M_2))$ and $\mathscr{R}(N) = Cut_n(\mathscr{R}(M_1'), \mathscr{R}(M_2))$ follows by definition of $>_{rpo}$. □

**Lemma 6.10.** $\mathscr{F} \subseteq \mathscr{SN}_{\mathscr{TPC}_{ES}}$.

**Proof.** Let $M \in \mathscr{F}$. Suppose that there exists an infinite reduction sequence:

$$M \longrightarrow_{TPC_{ES}} M_1 \longrightarrow_{TPC_{ES}} M_2 \longrightarrow_{TPC_{ES}} M_3 \ldots.$$

Since $M \in \mathscr{F}$ and $\mathscr{F}$ is closed by reduction (Lemma 6.6) all the terms in the infinite sequence are in $\mathscr{F}$ so that we can construct an infinite sequence:

$$\mathscr{R}(M) >_{rpo} \mathscr{R}(M_1) >_{rpo} \mathscr{R}(M_2) >_{rpo} \mathscr{R}(M_3) \ldots$$

which leads to a contradiction since $>_{rpo}$ is well-founded. □

The above result is used to prove PSN and strong normalization for $TPC_{ES}$:

**Theorem 6.11** (PSN for $TPC_{\mathrm{ES}}$). *$\longrightarrow_{TPC_{\mathrm{ES}}}$ preserves $\longrightarrow_{TPC}$-strong normalization of TPC raw terms.*

**Proof.** If $M \in \mathscr{T}_{TPC_{\mathrm{ES}}}$ and $M \in SN_{TPC}$, then $M \in \mathscr{F}$. Hence, by Lemma 6.10. $M \in SN_{TPC_{\mathrm{ES}}}$. □

The result of $TPC_{\mathrm{ES}}$-strong normalization can finally be established by means of the following lemma which makes the bridge between typing derivations in $TPC_{\mathrm{ES}}$ and those in $TPC$.

**Lemma 6.12.** *If $\Gamma \rhd_{TPC_{\mathrm{ES}}} M : A$ is provable, then $\Gamma \rhd_{TPC} ES(M) : A$ is provable.*

**Proof.** If $\Gamma \rhd_{TPC_{\mathrm{ES}}} M : A$, then by Theorem 3.7 $\Gamma \rhd_{TPC_{\mathrm{ES}}} ES(M) : A$. Since $ES(M)$ is a pure term by Lemma 6.1, then the rules to type terms with substitutions are not used and therefore $\Gamma \rhd ES(M) : A$ in $TPC$. □

**Theorem 6.13** (Strong normalization for $TPC_{\mathrm{ES}}$). *If $\Gamma \rhd_{TPC_{\mathrm{ES}}} M : A$, then $M \in SN_{TPC_{\mathrm{ES}}}$.*

**Proof.** Let $\Gamma \rhd_{TPC_{\mathrm{ES}}} M : A$. Then it is easy to see that every sub-term $N$ of $M$ is typable. By Lemma 6.12 for every sub-term $N$ of $M$ we have that $ES(N)$ is typable in $TPC$, and thus, by Theorem 5.21, we also have that $ES(N)$ is in $SN_{TPC}$. This means that $M$ is in $\mathscr{F}$ and so $M \in SN_{TPC_{\mathrm{ES}}}$ by Lemma 6.10. □

Even if explicit substitutions can always be eliminated (Lemma 6.1) , it is not true, in general, that the $TPC_{\mathrm{ES}}$ normal form of a term does not contain the *let* constructor. Indeed, the term $\mathtt{let}\ x\ \mathtt{be}\ \langle y, v\rangle\ \mathtt{in}\ z$ contains a *let* constructor and is in $TPC_{\mathrm{ES}}$ normal form. This is quite natural since pattern matching cannot be performed on terms that are not sufficiently specified or instantiated.

Say that a closed term is a *Value* when it is in $TPC_{\mathrm{ES}}$ normal form and it has one of the following forms: $\langle M, M\rangle$, $\mathtt{inl}_A(M)$,$\mathtt{inr}_A(M)$, $\lambda P : A.M$. A natural question arises: is it possible to evaluate any given (functional) *program* (modeled by a $TPC_{\mathrm{ES}}$ *closed* term), to a *result* (modeled by a value)? The answer to such a question is positive. That is, we have the following result:

**Theorem 6.14** (Reduction to values for closed well-typed terms). *Let $M$ be a* closed *and well-typed term of $TPC_{\mathrm{ES}}$. Then $M$ reduces to a value. In particular*:

1. *If $M$ has a product type, then it reduces to a pair.*
2. *If $M$ has a functional type, then it reduces to a lambda abstraction.*
3. *If $M$ has a sum type, then it reduces to an injection.*

**Proof.** By Lemma 6.1, we can suppose that $M$ is pure. Since $M$ is a typed term, then it is $TPC_{\mathrm{ES}}$-strongly normalizing by Theorem 6.13. Then the proof proceeds by induction on the maximal length of a $TPC_{\mathrm{ES}}$-reduction sequence starting at $M$. For the base case, let us suppose that $M$ is in $TPC_{\mathrm{ES}}$-normal form. In this case we can reason by induction on $M$.

1. The case where $M$ is a variable is not possible since variables are not closed terms.
2. If $M$ is a pair, or a lambda abstraction, or an $\texttt{inl}()$ or an $\texttt{inr}()$, then the result trivially holds.
3. The case $M = [M \mid_\xi M]$ is ruled out by the fact that such term is not closed.
4. The case $M = z \texttt{ of } M \texttt{ is } P \texttt{ in } N$ is not possible since such a term would not be not closed.
5. The case $M = \lambda P : A.M \texttt{ of } M \texttt{ is } P : A \texttt{ in } M$ is not possible because such a term would not be normal.
6. If $M = \texttt{let } R \texttt{ be } P \texttt{ in } L$, then the fact that $M$ is in normal form leaves the following possible cases (according to the nature of the pattern $P$):
   - $P = \langle P_1, P_2 \rangle$ and $R$ is not a pair. Since $R$ is also closed and typed, then by induction hypothesis $R$ reduces to a pair and so $M$ is not in normal form, which leads to a contradiction.
   - $P = \sharp z$ and $R$ is not an abstraction. Since $R$ is also closed and typed, then by induction hypothesis $R$ reduces to an abstraction and so $M$ is not in normal form, which leads to a contradiction.
   - $P = (P_1 \mid_\xi P_2)$ and $R$ is neither $\texttt{inl}(S)$ nor $\texttt{inr}(S)$. Since $S$ is also closed and typed, then by induction hypothesis $S$ reduces to an $\texttt{inl}()$ or to an $\texttt{inr}()$ and so $M$ is not in normal form, which leads to a contradiction.
   - $P = x$ or $P = \sharp x$. This case is ruled out by the fact that $M$ is normal form.
   - The case where $P$ is a contraction pattern is similar to the above one.
   - The case where $P$ and $R$ have a base type is ruled out because if $M$ is closed, $R$ must be closed, and no closed term may have a base type, as it can easily be shown by induction on the term.

If $M$ is not in $TPC_{\mathrm{ES}}$-normal form, then there is $N$ such that $M \longrightarrow_{TPC} N$. Now, the maximal length of a $TPC_{\mathrm{ES}}$-reduction sequence starting at $N$ is strictly smaller than the maximal length of a $TPC_{\mathrm{ES}}$-reduction sequence starting at $M$. Also, $N$ is necessarily closed, so that we can conclude by i.h that the property holds. □

## 7. Conclusion

This paper proposes a rational reconstruction of pattern matching via cut elimination. The main ideas and results can be summarized by the following points:

- A typed pattern calculus $TPC_{\mathrm{ES}}$ is proposed, where both typing and reduction are modeled on sequent calculus in the spirit of the Curry–Howard isomorphism.
- Pattern matching and substitution computations are *explicitly* coded in $TPC_{\mathrm{ES}}$ and correspond to sequences of cut elimination steps in sequent proofs.
- The calculus enjoys those fundamental properties required by any well-behaved calculus, such as: subject reduction, confluence, preservation of strong normalization w.r.t. the system $TPC$, strong normalization and reduction to values for closed terms.

Our calculus is a major step forward with respect to the other proposals that have been made these last years in a similar direction. Indeed, there are some term

assignments for the same intuitionistic sequent system we deal with, which can be roughly classified in the following categories:

- Neither patterns nor explicit substitutions are considered [2],
- Only simple patterns (non-nested) are taken into account [18,38],
- No patterns are used but cut elimination models explicit substitutions [16,22,37],
- Nested patterns are used and only some reduction rules, not including explicit substitutions, are suggested [30].

Now, let us consider where the present work might be improved, by further research.

To start with, the restriction imposed on the set of free communication variables for the typing rules $(+left)$, $(app)$, $(let)$, $(sub*)$ may seem unnecessarily drastic, since its effect is to severally limit the set of logical sequent proofs having a correspondent in proofs of the typing system for $TPC_{\mathrm{ES}}$. A milder restriction on these rules could be studied so that future work is required on this issue.

Also, the *app* typing rule of $TPC_{\mathrm{ES}}$ system can look weird, from a logical point of view, since, as we already observed, the logical counterpart of this typing rule is a *derived* rule in the sequent calculus for intuitionist minimal logic. The motivation to have such a typing rule as primitive is the need to close the set of typable terms with respect to $TPC_{\mathrm{ES}}$ reduction. However, as Roy Dyckhoff pointed out, such a "strange" typing rule might be dropped, but this would require to replace the reduction rules $(Of)$, $(Dapp1)$ and $(Dapp3)$ by a reduction rule like

$$(z \texttt{ of } M \texttt{ is } P \texttt{ in } N)[z/\lambda Q.J] \longrightarrow \texttt{let } (\texttt{let } M \texttt{ be } P \texttt{ in } J) \texttt{ be } Q \texttt{ in } N.$$

However, this idea has a major inconvenient which is the loss of a natural semantics for the explicit substitution constructor $[\_/\_]$. Indeed, within our reduction system we are able to show a useful feature of explicit substitutions: the $ES$-normal form of a term $M[x/N]$ is just the $ES$-normal form of $M$ where $x$ is replaced by the $ES$-normal form of $N$ (Lemma 2.7). This property would be completely lost if the above modification were done.

Even if $TPC_{\mathrm{ES}}$ is equipped with explicit pattern matching and explicit substitutions, there is still one operation which has remained at the meta-level, namely sum replacement (Section 2.3). However, in a real implementation also this operations needs to be explicitly encoded. A subject of future investigation might be the complete formulation of $TPC_{\mathrm{ES}}$ by means of explicit constructors. Another important issue is the study of reduction notions and evaluation strategies used in real programming languages with pattern matching features as well as the introduction of richer types such as for example polymorphic types or dependent types.

Since there is a deep connection between $S4$ intuitionist modal logic and LISP's `eval` and `quote` primitives in functional programming [13,19], it would be interesting to study whether our formalism might be extended to that logic.

## Acknowledgements

We are very grateful to Julien Forest who pointed out a mistake on an earlier version of this work. We also wish to thank Roy Dyckhoff and Jean Goubault-Larrecq, for their

careful reading and their criticisms. We would like to thank Val Tannen for valuable preliminary discussions concerning cut elimination and patterns, and for his interesting remarks. Finally, we would like to thank the anonymous referees of [10] for their contribution to the improvement of the presentation of this work.

## Appendix

**Lemma 3.5.** *If* $\Gamma \triangleright M : A$, *then* $\Gamma_{(\xi,\mathtt{K})} \triangleright M\{\xi \leftarrow \mathtt{K}\} : A$.

**Proof.** The proof is by induction on the height $h$ of the proof $\mathscr{P}$ of the judgment $\Gamma \triangleright M : A$.

If $h = 0$ then $\mathscr{P}$ is an axiom, so that $M$ is a usual variable and all the patterns in $\Gamma$ are variables. In this case, $\Gamma_{(\xi,\mathtt{K})} \triangleright M\{\xi \leftarrow \mathtt{K}\} : A$ coincides with $\Gamma \triangleright M : A$.

If $h > 0$, different cases must be syconsidered, according to the last rule in $\mathscr{P}$.

- The last rule is a $\times right$, applied to the judgments $\Gamma \triangleright M_1 : A_1$ and $\Gamma \triangleright M_2 : A_2$ (hence $M$ is $\langle M_1, M_2 \rangle$ and $A$ is $\langle A_1, A_2 \rangle$). By the induction hypothesis we have proofs of $\Gamma_{(\xi,\mathtt{K})} \triangleright M_1\{\xi \leftarrow \mathtt{K}\} : A_1$ and $\Gamma_{(\xi,\mathtt{K})} \triangleright M_2\{\xi \leftarrow \mathtt{K}\} : A_2$. Since $M\{\xi \leftarrow \mathtt{K}\}$ is $\langle M_1\{\xi \leftarrow \mathtt{K}\}, M_2\{\xi \leftarrow \mathtt{K}\} \rangle$, it suffices to apply the $\times right$ rule to $\Gamma_{(\xi,\mathtt{K})} \triangleright M_1\{\xi \leftarrow \mathtt{K}\} : A_1$ and $\Gamma_{(\xi,\mathtt{K})} \triangleright M_2\{\xi \leftarrow \mathtt{K}\} : A_2$ to get the result.
- The last rule is a $\times left$, applied to $P : B, Q : C, \Gamma' \triangleright M : A$. By the induction hypothesis we have a proof of $(P : B, Q : C, \Gamma')_{(\xi,\mathtt{K})} \triangleright M\{\xi \leftarrow \mathtt{K}\} : A$, i.e. of $P : B_{(\xi,\mathtt{K})}$, $Q : C_{(\xi,\mathtt{K})}, \Gamma'_{(\xi,\mathtt{K})} \triangleright M\{\xi \leftarrow \mathtt{K}\} : A$. Since $(\langle P, Q \rangle : B \times C, \Gamma')_{(\xi,\mathtt{K})}$ is $\langle P_{(\xi,\mathtt{K})}, Q_{(\xi,\mathtt{K})} \rangle : B \times C, \Gamma'_{(\xi,\mathtt{K})}$, it suffices to apply the $\times left$ rule to $(P : B, Q : C, \Gamma')_{(\xi,\mathtt{K})} \triangleright M\{\xi \leftarrow \mathtt{K}\} : A$ to get the result.
- The last rule is a $+right1$, applied to $\Gamma \triangleright M_1 : B$ (so that $A$ is $B + C$ and $M$ is $inl_C(M_1)$) or is a $+right2$ applied to $\Gamma \triangleright M_1 : C$ (so that $A$ is $B + C$ and $M$ is $inr_B(M_1)$). In both cases, the proof is a straightforward application of the induction hypothesis, as in the previous cases.
- The last rule is a $+left$, applied to $P : B, \Gamma' \triangleright M_1 : A$ and $Q : C, \Gamma' \triangleright M_2 : A$, so that $\Gamma$ is $(P \mid_\psi Q) : B + C, \Gamma'$ and $M$ is $[M_1 \mid_\psi M_2]$. Note that the pattern type assignment $(P \mid_\psi Q) : B + C, \Gamma'$ is linear, hence $\psi$ occurs just once in it. We need to distinguish two sub-cases.
  1. $\psi = \xi$. Since $\xi$ does not occur in $P$, $Q$, $\Gamma'$, in this case the judgment that we need to prove is

     $$P : B, \Gamma' \triangleright M_1\{\xi \leftarrow \mathtt{L}\} : A, \quad \text{if } \mathtt{K} = \mathtt{L},$$

     $$Q : C, \Gamma' \triangleright M_2\{\xi \leftarrow \mathtt{R}\} : A, \quad \text{if } \mathtt{K} = \mathtt{R}.$$

     Let us consider the case where $\mathtt{K} = \mathtt{L}$ (the other one is similar). By applying the induction hypothesis to the proof of $P : B, \Gamma' \triangleright M_1 : A$ and using the fact that $\xi$ does not occur in $P : B, \Gamma'$ (so that $P_{(\xi,\mathtt{K})} = P$), we get a proof of $P : B, \Gamma' \triangleright M_1\{\xi \leftarrow \mathtt{L}\} : A$.

2. $\psi \neq \xi$. In this case, the judgment that we need to prove is

$$(P_{(\xi,\mathtt{K})} \mid_{\psi} Q_{(\xi,\mathtt{K})}) : B + C, \Gamma'_{(\xi,\mathtt{K})} \rhd [M_1\{\xi \leftarrow \mathtt{K}\} \mid_{\psi} M_2\{\xi \leftarrow \mathtt{K}\}] : A.$$

Applying the induction hypothesis to the proofs of $P : B, \Gamma' \rhd M_1 : A$ and $Q : C, \Gamma' \rhd M_2 : A$, we get proofs of $P_{(\xi,\mathtt{K})} : B, \Gamma'_{(\xi,\mathtt{K})} \rhd M_1\{\xi \leftarrow \mathtt{K}\}$ and $Q_{(\xi,\mathtt{K})} : C, \Gamma'_{(\xi,\mathtt{K})} \rhd M_2\{\xi \leftarrow \mathtt{K}\}$. An application of the $+left$ rule to these last judgments gives the desired result.

- The last rule is a $\rightarrow right$, applied to $P : B, \Gamma' \rhd M_1 : C$, so that $A$ is $B \rightarrow C$ and $M$ is $\lambda P.M_1$. By $\alpha$-conversion, we may assume that $\xi$ does not occur in $P$ so that $P_{(\xi,\mathtt{K})} = P$. Thus, by the induction hypothesis we have a proof of $P : B, \Gamma'_{(\xi,\mathtt{K})} \rhd M_1\{\xi \leftarrow \mathtt{K}\} : C$. An application of the $\rightarrow right$ rule to such a judgment gives a proof of $\Gamma_{(\xi,\mathtt{K})} \rhd \lambda P : B.M_1\{\xi \leftarrow \mathtt{K}\} : B \rightarrow C$, i.e. the desired result since $\lambda P : B.M_1\{\xi \leftarrow \mathtt{K}\}$ is exactly $(\lambda P : B.M_1)\{\xi \leftarrow \mathtt{K}\}$.
- The last rule is a $\rightarrow left$, applied to $\Gamma' \rhd M_1 : B$ and $Q : C, \Gamma' \rhd N : A$, so that $M$ is $z$ `of` $M_1$ `is` $Q : B$ `in` $N$. By $\alpha$-conversion, we may assume that $\xi$ does not occur in $Q$ so that $Q_{(\xi,\mathtt{K})} = Q$. Thus, by the induction hypothesis we have a proof of $\Gamma'_{(\xi,\mathtt{K})} \rhd M_1\{\xi \leftarrow \mathtt{K}\} : B$ and $Q : C, \Gamma'_{(\xi,\mathtt{K})} \rhd N\{\xi \leftarrow \mathtt{K}\} : A$. An application of the $\rightarrow left$ rule to such a judgment provides a proof of

$$\sharp z : B \rightarrow C, \Gamma'_{(\xi,\mathtt{K})} \rhd (z \ \mathtt{of}\ M_1\ \mathtt{is}\ Q : B\ \mathtt{in}\ N)\{\xi \leftarrow \mathtt{K}\} : A,$$

i.e. the desired result.
- The last rule is a *let*, applied to $\Gamma \rhd M_1 : B$ and $P : B, \Gamma \rhd N : A$, so that $M$ is `let` $M_1$ `be` $P : B$ `in` $N : A$. Again, we may suppose that $P$ does not contain $\xi$. By the induction hypothesis we have a proof of $\Gamma_{(\xi,\mathtt{K})} \rhd M_1\{\xi \leftarrow \mathtt{K}\} : B$ and $P : B, \Gamma_{(\xi,\mathtt{K})} \rhd N\{\xi \leftarrow \mathtt{K}\} : A$. Hence, an application of the *let* rule gives a proof of

$$\Gamma_{(\xi,\mathtt{K})} \rhd \mathtt{let}\ M_1\{\xi \leftarrow \mathtt{K}\}\ \mathtt{be}\ P : B\ \mathtt{in}\ N\{\xi \leftarrow \mathtt{K}\} : A,$$

i.e. the desired result.
- The last rule is a $sub*$, applied to $\Gamma \rhd M_1 : B$ and $?v : B, \Gamma \rhd N : A$, so that $M$ is $M_1[v/N]$. By the induction hypothesis we have proofs of $\Gamma_{(\xi,\mathtt{K})} \rhd M_1\{\xi \leftarrow \mathtt{K}\} : B$ and $?v, \Gamma_{(\xi,\mathtt{K})} \rhd N\{\xi \leftarrow \mathtt{K}\} : A$. An application of the $sub*$ rule provides a proof of

$$\Gamma_{(\xi,\mathtt{K})} \rhd M_1\{\xi \leftarrow \mathtt{K}\}[v/N\{\xi \leftarrow \mathtt{K}\}],$$

i.e. the desired result.
- The last rule is *layered*. Then $\Gamma = @(P,Q) : B, \Gamma'$ where $@(P,Q)$ is the pattern introduced by the layered rule and by induction hypothesis we know that $(P : B, Q : B, \Gamma')_{(\xi,\mathtt{K})} \rhd M\{\xi \leftarrow \mathtt{K}\} : A$ is provable. Since $(P : B, Q : B, \Gamma')_{(\xi,\mathtt{K})} = (P : B)_{(\xi,\mathtt{K})}, (Q : B)_{(\xi,\mathtt{K})}, \Gamma'_{(\xi,\mathtt{K})}$ then it is sufficient to apply the *layered* rule to such a judgment to obtain $@((P : B)_{(\xi,\mathtt{K})}, (Q : B_{(\xi,\mathtt{K})})), \Gamma'_{(\xi,\mathtt{K})} \rhd M\{\xi \leftarrow \mathtt{K}\} : A$, that is $(@(P,Q) : B, \Gamma')_{(\xi,\mathtt{K})} \rhd M\{\xi \leftarrow \mathtt{K}\} : A$.
- The last rule is *wildcard*. Then $\Gamma = P : B, \Gamma'$ where $P$ is the pattern introduced by the *wildcard* rule and by induction hypothesis we know that $\Gamma'_{(\xi,\mathtt{K})} \rhd M\{\xi \leftarrow \mathtt{K}\} : A$

is provable. We have then two cases to consider:
- $\xi \in FV(P)$. Then, since pattern assignments are linear we have $\xi \notin Var(\Gamma')$ and $\Gamma'_{(\xi,\mathtt{K})} = \Gamma'$. Let $Q = P_{(\xi,\mathtt{K})}$. Since $P : B, \Gamma'$ is linear and $Var(Q) \subseteq Var(P)$, then $Q : B, \Gamma'$ is also linear. Moreover since $P \in AP(B)$, then by Lemma 3.4 $Q \in AP(C)$ for some $C$, so that $Q : C, \Gamma'_{(\xi,\mathtt{K})} \triangleright M\{\xi \leftarrow \mathtt{K}\} : A$ is derivable from $\Gamma'_{(\xi,\mathtt{K})} \triangleright M\{\xi \leftarrow \mathtt{K}\} : A$ by the *wildcard* rule, which shows the property for this case.
- $\xi \notin FV(P)$. In this case $P_{(\xi,\mathtt{K})} = P$ and so a straightforward application of the induction hypothesis followed by an application of the *wildcard* rule suffices.

- The last rule is $app$, applied to $\Gamma \triangleright \lambda P.J : B \to C$, $\Gamma \triangleright M_1 : B$ and $\Gamma, Q : C \triangleright N : A$, so that $M$ is $\lambda P.J$ `of` $M_1$ `is` $Q$ `in` $N$. We may suppose that $\xi$ does not occur in $Q$ by $\alpha$-conversion. The induction hypothesis gives proofs of:

$$\Gamma_{(\xi,\mathtt{K})} \triangleright (\lambda P.J)\{\xi \leftarrow \mathtt{K}\} : B \to C \qquad \Gamma_{(\xi,\mathtt{K})} \triangleright M_1\{\xi \leftarrow \mathtt{K}\} : B$$
$$\Gamma_{(\xi,\mathtt{K})}, Q : C \triangleright N\{\xi \leftarrow \mathtt{K}\} : A.$$

Hence, an application of the $app$ rule to the above judgments gives the desired result. □

**Lemma 3.6** (Loosening). *Let $\Gamma, P : A \triangleright M : B$ be a provable judgment. We have*:
1. *If $P = (P_1 \mid_\xi P_2)$ and $\xi \notin FV(M)$ then $\Gamma \triangleright M : B$ is provable.*
2. *If $Var(P) \cap FV(M) = \emptyset$, then $\Gamma \triangleright M : B$ is provable.*

**Proof.** By induction on the height $h$ of the proof $\mathscr{P}$ of $\Gamma, P : A \triangleright M : B$.

*Base case*: $h = 0$. The proof $\mathscr{P}$ is an axiom. Then, $M$ is a variable $x_i$. The first item of the lemma is vacuously true because $\xi$ cannot occur in $P$. For the second item, it suffices to observe that if $Var(P) \cap FV(M) = \emptyset$, then $P$ must be *another* variable $x_j$, so that $\mathscr{P}$ has the form:

$$\Gamma, x_j : A_j \triangleright x_i : A_i,$$

where $x_i$ occurs in $\Gamma$. Clearly, $\Gamma \triangleright x_i : A_i$ is still an axiom.

*Inductive Step*: $h > 0$. We take as the induction hypothesis that both items of the lemma are true for any proof having a height $h'$ such that $h' < h$. Several cases, corresponding to the last inference rule in $\mathscr{P}$, need to be considered.

1. The last inference in $\mathscr{P}$ is a $\times$*right*-rule:

$$\frac{\Gamma, P : A \triangleright M_1 : A_1 \qquad \Gamma, P : A \triangleright M_2 : A_2}{\Gamma, P : A \triangleright \langle M_1, M_2 \rangle : A_1 \times A_2} \ (\times right)$$

   hence $M$ is $\langle M_1, M_2 \rangle$. Under the respective hypotheses of both the items of the lemma, we can apply the induction hypothesis to the proofs of $\Gamma, P : A \triangleright M_1 : A_1$ and $\Gamma, P : A \triangleright M_2 : A_2$. Hence, we have proofs of $\Gamma \triangleright M_1 : A_1$ and $\Gamma \triangleright M_2 : A_2$, that we can combine via an application of the $\times$*right* rule to get $\Gamma \triangleright \langle M_1, M_2 \rangle : A_1 \times A_2$.
2. The last inference in $\mathscr{P}$ is a $+$*left*.

In both cases, the sum pattern introduced by such an inference must be a pattern $(Q_1 \mid_\psi Q_2) : D + C$ where $\psi$ is a communication variable different from the $\xi$ occurring in $P = (P_1 \mid_\xi P_2)$, otherwise $\xi$ would be free in $M$. Thus, $M$ is $[M_1 \mid_\psi M_2]$ and the $+left$ is

$$\frac{Q_1 : D, \Gamma' \triangleright M_1 : B \qquad Q_2 : C, \Gamma' \triangleright M_2 : B}{(Q_1 \mid_\psi Q_2) : D + C, \Gamma' \triangleright [M_1 \mid_\psi M_2] : B} \quad (+left)$$

where $\Gamma = (Q_1 \mid_\psi Q_2) : D + C, \Gamma''$ and $\Gamma' = P : A, \Gamma''$. Since $\xi$ is not free neither in $M_1$ nor in $M_2$, by induction hypothesis we have proofs of $Q_1 : D, \Gamma'' \triangleright M_1 : B$ and $Q_2 : C, \Gamma'' \triangleright M_2 : B$. Thus, via a $+left$ rule, we get a proof of $(Q_1 \mid_\psi Q_2) : B + C, \Gamma'' \triangleright [M_1 \mid_\psi M_2] : B$, that is a proof of $\Gamma \triangleright M : B$.

3. All the other cases are treated in a similar way. □

**Lemma 5.3.** *The reflexive transitive closures of* $\gg$ *and* $\longrightarrow_{TPC}$ *are the same relation.*

**Proof.** To show the inclusion $\gg^* \subseteq \longrightarrow^*_{TPC}$, we show that $\gg \subseteq \longrightarrow^*_{TPC}$ by induction on the definition of $\gg$.

- If $M \gg M$, then $M \longrightarrow^*_{TPC} M$ trivially holds.
- In the case where $M \gg M'$ comes from internal reductions, the property holds by i.h. and the fact that $\longrightarrow^*_{TPC}$ is closed for all contexts.
- If $\texttt{let}\ M\ \texttt{be}\ \_\ \texttt{in}\ N \gg N'$, then $\texttt{let}\ M\ \texttt{be}\ \_\ \texttt{in}\ N \longrightarrow_{TPC} N \longrightarrow^*_{TPC} N'$, where the reduction sequence $N \longrightarrow^*_{TPC} N'$ holds by induction hypothesis.
- If $\texttt{let}\ M\ \texttt{be}\ x\ \texttt{in}\ N \gg N'\{x \leftarrow M'\}$, then $M \longrightarrow^*_{TPC} M'$ and $N \longrightarrow^*_{TPC} N'$ by i.h, thus we have $M\{x \leftarrow N\} \longrightarrow^*_{TPC} M'\{x \leftarrow N\} \longrightarrow^*_{TPC} M'\{x \leftarrow N'\}$ by Lemmas 5.2 and 5.2.
- If $M$ is a $\lambda$-abstraction and $\texttt{let}\ M\ \texttt{be}\ \sharp z\ \texttt{in}\ N \gg N'\{z \leftarrow M'\}$, the proof is similar to the previous case.
- If $\texttt{let}\ \texttt{inl}(M)\ \texttt{be}\ (P_1 \mid_\xi P_2)\ \texttt{in}\ N \gg \texttt{let}\ M'\ \texttt{be}\ P_1\ \texttt{in}\ N'\{\xi \leftarrow \texttt{L}\}$ where $M \gg M'$ and $N \gg N'$, then we have $M \longrightarrow^*_{TPC} M'$ and $N \longrightarrow^*_{TPC} N'$ by i.h. As a consequence we have by definition that $\texttt{let}\ \texttt{inl}(M)\ \texttt{be}\ (P_1 \mid_\xi P_2)\ \texttt{in}\ N$ reduces to $\texttt{let}\ M\ \texttt{be}\ P_1\ \texttt{in}\ N\{\xi \leftarrow \texttt{L}\}$, which reduces to the term $\texttt{let}\ M'\ \texttt{be}\ P_1\ \texttt{in}\ N'\{\xi \leftarrow \texttt{L}\}$ by Lemma 5.2. The same happens with $\texttt{let}\ \texttt{inr}(M)\ \texttt{be}\ (P_1 \mid_\xi P_2)\ \texttt{in}\ N \gg \texttt{let}\ M'\ \texttt{be}\ P_2\ \texttt{in}\ N'\{\xi \leftarrow \texttt{R}\}$.
- If $P = \langle P_1, P_2 \rangle$ or $P = @(P_1, P_2)$, then the property holds by i.h and the fact that reduction is closed under any context.
- If $(\lambda P.M)\ \texttt{of}\ N\ \texttt{is}\ Q\ \texttt{in}\ L \gg \texttt{let}\ (\texttt{let}\ N'\ \texttt{be}\ P\ \texttt{in}\ M')\ \texttt{be}\ Q\ \texttt{in}\ L'$, where $M \gg M'$, $N \gg N'$, $L \gg L'$, then by i.h. we have that $M \longrightarrow^*_{TPC} M'$, $N \longrightarrow^*_{TPC} N'$ and $L \longrightarrow^*_{TPC} L'$ so

$$\begin{array}{l}(\lambda P.M)\ \texttt{of}\ N\ \texttt{is}\ Q\ \texttt{in}\ L \longrightarrow^*_{TPC} \\ (\lambda P.M')\ \texttt{of}\ N'\ \texttt{is}\ Q\ \texttt{in}\ L' \longrightarrow_{TPC} \\ \texttt{let}\ (\texttt{let}\ N'\ \texttt{be}\ P\ \texttt{in}\ M')\ \texttt{be}\ Q\ \texttt{in}\ L'\end{array}$$

To show the inclusion $\longrightarrow^*_{TPC} \subseteq \gg^*$ we show that $\longrightarrow_{TPC} \subseteq \gg$ by induction on the definition of $\longrightarrow_{TPC}$.

- In the case where $M \longrightarrow_{TPC} M'$ comes from internal reductions, the property holds by i.h. and the fact that $\gg$ is closed for all contexts.
- In the case where $\longrightarrow_{TPC}$ comes from an external reduction, then the property follows by definition of $\gg$ and the fact that $M \gg M$ holds for every term $M$. □

**Lemma 5.5.** *If $M \gg M'$, then there is $N$ such that $M\{\xi \leftarrow \mathtt{K}\} \gg N$ and $M'\{\xi \leftarrow \mathtt{K}\} \gg N$.*

**Proof.** By induction on $M$ and by cases.

1. In the base case, $M$ is a variable and the property trivially holds.
2. In the inductive case, several cases need to be considered.
   - If $M \gg M = M'$, the property trivially holds, exactly as in the base case.
   - If $M \gg M'$ corresponds to one of the cases $\mathtt{inl}(L) \gg \mathtt{inl}(L')$, $\mathtt{inr}(L) \gg \mathtt{inr}(L')$, or $\lambda P.L \gg \lambda P.L'$, where $L \gg L'$, then the property directly follows by induction hypothesis.
   - If $M \gg M'$ corresponds to $\mathtt{let}\ \mathtt{inl}(L)\ \mathtt{be}\ (P_1 \mid_\rho P_2)\ \mathtt{in}\ K \gg \mathtt{let}\ L'\ \mathtt{be}\ P_1\ \mathtt{in}\ K\{\rho \leftarrow \mathtt{L}\}$, where $L \gg L'$, then by i.h. there is $N$ such that $L\{\xi \leftarrow \mathtt{K}\} \gg N$ and $L'\{\xi \leftarrow \mathtt{K}\} \gg N$. Since $\rho$ is bound in $M$, then by $\alpha$ conversion we can assume $\rho$ different from $\xi$. By Definition of $\gg$ we have

$$\begin{aligned}&\mathtt{let}\ \mathtt{inl}(L\{\xi \leftarrow \mathtt{K}\})\ \mathtt{be}\ (P_1 \mid_\xi P_2)\ \mathtt{in}\ K\{\xi \leftarrow \mathtt{K}\}\\ &\quad \gg \mathtt{let}\ \mathtt{inl}(N)\ \mathtt{be}\ (P_1 \mid_\xi P_2)\ \mathtt{in}\ K\{\xi \leftarrow \mathtt{K}\}\\ &\quad \gg \mathtt{let}\ N\ \mathtt{be}\ P_1\ \mathtt{in}\ K\{\xi \leftarrow \mathtt{K}\}\{\rho \leftarrow \mathtt{L}\}\end{aligned}$$

     and

$$\begin{aligned}&\mathtt{let}\ L'\{\xi \leftarrow \mathtt{K}\}\ \mathtt{be}\ P_1\ \mathtt{in}\ K\{\rho \leftarrow \mathtt{L}\}\{\xi \leftarrow \mathtt{K}\}\\ &\quad \gg \mathtt{let}\ N\ \mathtt{be}\ P_1\ \mathtt{in}\ K\{\rho \leftarrow \mathtt{L}\}\{\xi \leftarrow \mathtt{K}\}\\ &\quad = \mathtt{let}\ N\ \mathtt{be}\ P_1\ \mathtt{in}\ K\{\xi \leftarrow \mathtt{K}\}\{\rho \leftarrow \mathtt{L}\}\end{aligned}$$

     The case $\mathtt{let}\ \mathtt{inr}(M)\ \mathtt{be}\ (P_1 \mid_\rho P_2)\ \mathtt{in}\ N \gg \mathtt{let}\ M'\ \mathtt{be}\ P_2\ \mathtt{in}\ N\{\xi \leftarrow \mathtt{R}\}$ is similar.
   - If $M \gg M'$ comes from $\langle L, K\rangle \gg \langle L', K'\rangle$, then we have either $\mathtt{let}\ L\ \mathtt{be}\ P\ \mathtt{in}\ K \gg \mathtt{let}\ L'\ \mathtt{be}\ P\ \mathtt{in}\ K'$, or $\mathtt{let}\ L\ \mathtt{be}\ \_\ \mathtt{in}\ K \gg K'$ where $L \gg L'$ and $K \gg K'$. Then, the property follows by i.h.
   - If $M \gg M'$ comes from $[L \mid_\rho T] \gg [L' \mid_\rho T']$, where $L \gg L'$ and $T \gg T'$, then by i.h. there are $N_1$ and $N_2$ such that $L\{\xi \leftarrow \mathtt{K}\} \gg N_1$, $L'\{\xi \leftarrow \mathtt{K}\} \gg N_1$, $T\{\xi \leftarrow \mathtt{K}\} \gg N_2$ and $T'\{\xi \leftarrow \mathtt{K}\} \gg N_2$. There are three cases to consider. If $\rho = \xi$ and $\mathtt{K} = \mathtt{L}$ then, by Definition 2.2, we have $[L \mid_\rho T]\{\rho \leftarrow \mathtt{L}\} = L\{\rho \leftarrow \mathtt{L}\} \gg N_1$ and $[L' \mid_{\mathtt{L}} T']\{\rho \leftarrow \mathtt{L}\} = L'\{\rho \leftarrow \mathtt{L}\} \gg N_1$. The case $\rho = \xi$ and $\mathtt{K} = \mathtt{R}$ is similar.

     For the case $\rho \neq \xi$, we have $[L \mid_\rho T]\{\xi \leftarrow \mathtt{K}\} = [L\{\xi \leftarrow \mathtt{K}\} \mid_\rho ]T\{\xi \leftarrow \mathtt{K}\} \gg [N_1 \mid_\rho N_2]$ and $[L' \mid_\rho T']\{\xi \leftarrow K\} = [L'\{\xi \leftarrow \mathtt{K}\} \mid_\rho T'\{\rho \leftarrow \mathtt{K}\}] \gg [N_1 \mid_\rho N_2]$.

- If $M \gg M'$ comes from $\texttt{let}\ \langle L_1, L_2\rangle\ \texttt{be}\ \langle P_1, P_2\rangle\ \texttt{in}\ K \gg \texttt{let}\ L_1'\ \texttt{be}\ P_1\ \texttt{in}\ (\texttt{let}\ L_2'\ \texttt{be}\ P_2\ \texttt{in}\ K')$, where $\langle L_1, L_2\rangle \gg \langle L_1', L_2'\rangle$ and $K \gg K'$ the property follows by the induction hypothesis. The same holds for the case $\texttt{let}\ L\ \texttt{be}\ @(P_1)P_2\ \texttt{in}\ K \gg \texttt{let}\ L\ \texttt{be}\ P_1\ \texttt{in}\ (\texttt{let}\ L\ \texttt{be}\ P_2\ \texttt{in}\ K)$.
- If $M \gg M'$ comes from $\texttt{let}\ L\ \texttt{be}\ x\ \texttt{in}\ K \gg K'\{x \leftarrow L'\}$, where $L \gg L'$ and $K \gg K'$, then by i.h. there are $N_1$ and $N_2$ such that $L\{\xi \leftarrow \texttt{K}\} \gg N_1$, $L'\{\xi \leftarrow \texttt{K}\} \gg N_1$, $K\{\xi \leftarrow \texttt{K}\} \gg N_2$ and $K'\{\xi \leftarrow \texttt{K}\} \gg N_2$. We then have $\texttt{let}\ L\{\xi \leftarrow \texttt{K}\}\ \texttt{be}\ x\ \texttt{in}\ K\{\xi \leftarrow \texttt{K}\} \gg \texttt{let}\ N_1\ \texttt{be}\ x\ \texttt{in}\ N_2 \gg N_2\{x \leftarrow N_1\}$ and $K'\{x \leftarrow L'\}\{\xi \leftarrow \texttt{K}\} = K'\{\xi \leftarrow \texttt{K}\}\{x \leftarrow L'\{\xi \leftarrow \texttt{K}\}\} \gg N_2\{x \leftarrow N_1\}$ by Lemma 5.4. The case $\texttt{let}\ L\ \texttt{be}\ \sharp z\ \texttt{in}\ K \gg K'\{z \leftarrow L'\}$ is similar.
- For all the remaining cases, it is easy to see that the property holds by the induction hypothesis. □

**Lemma 5.11.** (1) *Every stable term $M$ of type $A$ is strongly normalizing.*

(2) *A term $xK_1 \ldots K_n$ of type $A$ is stable for arbitrary strongly normalizing terms $K_1 \ldots K_n$, where $n \geqslant 0$.*

**Proof.** The two properties can be shown at the same time by induction on the type $A$.

- If $A$ is an atomic type:
  1. By definition.
  2. By Lemma 5.10 the term $x\bar{K}$ is strongly normalizing, so $x\bar{K}$ is stable by definition.
- If $A$ is a product type:
  1. By definition.
  2. By Lemma 5.10 the term $x\bar{K}$ is strongly normalizing and the reduction sequences starting at $x\bar{K}$ can only proceed in the $K_i's$. Therefore, $x\bar{K}$ cannot reduce to a pair, and $x\bar{K}$ is stable by definition.
- If $A$ is a sum type:
  1. By definition.
  2. By Lemma 5.10 the term $x\bar{K}$ is strongly normalizing and the reduction sequences starting at $x\bar{K}$ can only proceed in the $K_i's$. Therefore, $x\bar{K}$ cannot reduce to an injection, and $x\bar{K}$ is stable by definition.
- If $A$ is a functional type:
  1. Let $A \equiv B \to C$ and let $x$ be a variable of type $B$. By the second induction hypothesis (with $n = 0$) $x : B$ is stable and by definition $Mx$ is of type $C$ and it is stable, hence, by the first induction hypothesis $Mx$ is strongly normalizing. Suppose now that $M$ is not strongly normalizing. Then there is an infinite reduction sequence $M \longrightarrow_{TPC} M_1 \longrightarrow_{TPC} \ldots$, thus an infinite reduction sequence $Mx \longrightarrow_{TPC} M_1x \longrightarrow_{TPC} \ldots$, which leads to a contradiction. Therefore $M$ is strongly normalizing.
  - Let $xK_1 \ldots K_n$ be of type $B \to C$ with all the $K_i's$ strongly normalizing. Let $N$ be any stable term of type $B$. From the first induction hypothesis $N$ is also strongly normalizing, by the second induction hypothesis $xK_1 \ldots K_nN$ is stable. Then, by definition, $xK_1 \ldots K_n$ is stable. □

**Lemma 5.13.** *If $M$ is a stable term and $M \longrightarrow_{TPC} N$, then $N$ is stable.*

**Proof.** We show the property by induction on the type $A$ of $M$. Note that $N$ is also of type $A$ by Theorem 5.1.

- *$A$ is a base type*: if $M$ is stable, then $M$ is strongly normalizing by Lemma 5.11. Then also $N$ is strongly normalizing since every reduction sequence starting at $N$ can be embedded in a reduction sequence starting at $M$, which terminates by hypothesis. We then conclude that $N$ is stable by definition.
- *$A$ is a product or a sum type*: as above we can conclude that $N$ is also strongly normalizing. On the other hand, when $N$ reduces to either a pair $\langle M_1, M_2\rangle$, an $\mathtt{inl}(M_1)$ or an $\mathtt{inr}(M_1)$, then also $M$ reduces to the same term and so $M_1$ and $M_2$ are stable because $M$ is stable. We then conclude that $N$ is stable.
- *$A$ is a functional type*: by definition of stability on arrow types, it suffices to show that $NL$ is stable for any stable term $L$. Now, given a stable $L$, $ML$ is stable because $M$ is so (by hypothesis), and $ML \longrightarrow_{TPC} NL$, thus by induction hypothesis $NL$ is stable. □

## References


[1] M. Abadi, L. Cardelli, P.L. Curien, J.-J. Lévy, Explicit substitutions, J. Funct. Program. 4 (1) (1991) 375–416.
[2] S. Abramsky, Computational interpretations of linear logic, Theoret. Comput. Sci. 111 (1993) 3–57.
[3] The Alfa WWW page. http://www.cs.chalmers.se/~hallgren/Alfa/.
[4] F. Baader, T. Nipkow, Term Rewriting and All That, Cambridge University Press, Cambridge, 1998.
[5] H. Barendregt, The Lambda Calculus: Its Syntax and Semantics, Studies in Logic and the Foundations of Mathematics, Revised Edition, vol. 103, North-Holland, Amsterdam, 1984.
[6] Z.-El-A. Benaissa, D. Briaud, P. Lescanne, J. Rouyer-Degli, $\lambda\upsilon$, a calculus of explicit substitutions which preserves strong normalisation, J. Funct. Program. 6 (5) (1996) 699–722.
[7] R. Bloo, K. Rose, Combinatory reduction systems with explicit substitution that preserve strong normalisation, in: H. Ganzinger (Ed.), 7th Internat. Conf. on Rewriting Techniques and Applications, Lecture Notes in Computer Science, vol. 1103, Springer, Berlin, July 1996.
[8] R. Burstall, D. MacQueen, D. Sanella, Hope: an experimental applicative language, in Proc. LISP Conference, Stanford University, Computer Science Department, July 1980, pp. 136–143.
[9] The Caml WWW page. http://caml.inria.fr/.
[10] S. Cerrito, D. Kesner, Pattern matching as cut elimination, in: G. Longo (Ed.), 14th Ann. IEEE Symp. on Logic in Computer Science (LICS), IEEE Computer Society Press, Silver Spring, MD, July 1999, pp. 98–108.
[11] H. Cirstea, C. Kirchner, $\rho$-calculus, the rewriting calculus, in: 5th Internat. Workshop on Constraints in Computational Logics, 1998.
[12] The Coq WWW page. http://pauillac.inria.fr/coq/index.html.
[13] R. Davies, F. Pfenning, A modal analysis of staged computation, in: Proc. 23th Ann. ACM Symp. on Principles of Programming Languages (POPL), ACM, New York, 1996, pp. 258–270.
[14] N. Dershowitz, Orderings for term rewriting systems, Theoret. Comput. Sci. 17 (3) (1982) 279–301.
[15] N. Dershowitz, J.-P. Jouannaud, Rewrite systems, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, vol. B, North-Holland, Amsterdam, 1990, pp. 243–309.
[16] R. Dyckhoff, C. Urban, Strong normalisation of Herbelin's explicit substitution calculus with substitution propagation, in: P. Lescanne (Ed.), Proc. 3rd Workshop on Explicit Substitutions: Theory and Applications to Programs and Proofs, June 2001, pp. 26–45.
[17] J. Gallier, Logic for Computer Science: Foundations of Automatic Theorem Proving, Wiley, New York, 1986.

[18] J. Gallier, Constructive logics part I: a tutorial on proof systems and typed $\lambda$-calculi, Theoret. Comput. Sci. 110 (2) (1993) 249–339.
[19] J. Goubault-Larrecq, On computational interpretations of the modal logic S4: cut elimination, 1996, available as http://theory.doc.ic.ac.uk:80/theory/guests/GoubaultJ/eq1.dvi.gz.
[20] C. Gunter, Semantics of Programming Languages: Structures and Techniques, Foundations of Computing Series, The MIT Press, Cambridge, MA, 1992.
[21] T. Hardin, Résultats de confluence pour les règles fortes de la logique combinatoire catégorique et liens avec les lambda-calculs, Thèse de Doctorat, Université de Paris VII, 1987.
[22] H. Herbelin, A $\lambda$-calculus structure isomorphic to sequent calculus structure, in: L. Pacholski, J. Tiuryn (Eds.), Proc. 8th Ann. Conf. of the European Association for Computer Science Logic (CSL), Lecture Notes in Computer Science, vol. 933, Springer, Berlin, September 1994.
[23] The HOL WWW page. http://www.dcs.gla.ac.uk/~tfm/fmt/hol.html.
[24] P. Hudak, S. Peyton-Jones, P. Wadler (Eds.), Report on the programming language Haskell, a non-strict, purely functional language (version 1.2). Sigplan Notices, 1992.
[25] J.-P. Jouannaud, M. Okada, Abstract data type systems, Theoret. Comput. Sci. 173 (1997) 349–391.
[26] S. Kamin, J.-J. Lévy, Attempts for generalizing the recursive path orderings. Handwritten paper, University of Illinois, 1980.
[27] D. Kesner, Confluence of extensional and non-extensional lambda-calculi with explicit substitutions, Theoret. Comput. Sci. 238 (1–2) (2000) 183–220.
[28] D. Kesner, L. Puel, V. Tannen, A typed pattern calculus, Inform. and Comput. 124 (1) (1996) 32–61.
[29] J.-L. Krivine, $\lambda$-calcul, types et modèles, Masson, Paris, 1990.
[30] Y. Lafont, Functional Programming and Linear Logic, Lecture Notes for the Summer School on Functional Programming and Constructive Logic, Glasgow University, Scotland, UK, 1989.
[31] The LEGO WWW page. http://www.dcs.ed.ac.uk/home/lego/.
[32] P.-A. Melliès, Typed $\lambda$-calculi with explicit substitutions may not terminate, in: M. Dezani-Ciancaglini, G. Plotkin (Eds.), Proc. 2nd Internat. Conf. of Typed Lambda Calculus and Applications, Lecture Notes in Computer Science, vol. 902, Springer, Berlin, April 1995.
[33] R. Milner, M. Tofte, R. Harper, The Definition of Standard ML, MIT Press, Cambridge, MA, 1990.
[34] The PVS WWW page. http://pvs.csl.sri.com/.
[35] D. Turner, Miranda: a non-strict functional language with polymorphic types, in: J.P. Jouannaud (Ed.), Proc. Conf. on Functional Programming Languages and Computer Architecture, Lecture Notes in Computer Science, vol. 201, Springer, Berlin, 1985, pp. 1–16.
[36] F. van Raamsdonk, Higher-order rewriting, in: P. Narendran, M. Rusinowitch (Eds.), 10th Internat. Conf. on Rewriting Techniques and Applications, Lecture Notes in Computer Science, vol. 1631, Springer, Berlin, July 1999, pp. 220–239.
[37] R. Vestergaard, J. Wells, Cut rules and explicit substitutions, Math. Struct. Comput. Sci. 11 (1) (2001) 131–168.
[38] P. Wadler, A Curry–Howard isomorphism for sequent calculus, Draft, 1993.