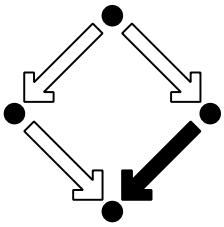
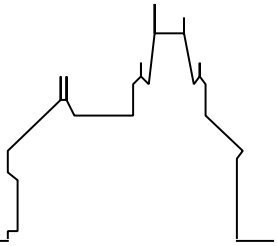


RISC-Linz
Research Institute for Symbolic Computation
Johannes Kepler University
A-4040 Linz, Austria, Europe



A Dynamic Pattern Calculus with Hedge Variables

Sandra Alves, Besik Dundua, Mário Florido, and
Temur Kutsia

RISC-Linz Report Series No. 12-20

Editors: RISC-Linz Faculty
K. Bosa, B. Buchberger, R. Hemmecke, T. Jebelean, M. Kauers, T. Kutsia,
G. Landsmann, F. Lichtenberger, P. Paule, V. Pillwein, N. Popov, H. Rolletschek,
J. Schicho, C. Schneider, W. Schreiner, W. Windsteiger, F. Winkler.

Supported by: the FCT fellowship (ref. SFRH/BD/62058/2009), the Austrian Science Fund (FWF) under the project SToUT (P 24087-N18), and the Rustaveli Science Foundation under the grant DI/16/4-120/11.

Copyright notice: Permission to copy is granted provided the title page is also copied.

A Dynamic Pattern Calculus with Hedge Variables

Sandra Alves[1], Besik Dundua[1,3], Mário Florido[1], and Temur Kutsia[2]

[1] DCC-FC & LIACC, University of Porto, Portugal
[2] RISC, Johannes Kepler University, Linz, Austria
[3] VIAM, Ivane Javakhishvili Tbilisi State University, Georgia

Abstract. The dynamic pattern calculus described in this paper integrates the functional mechanism of the lambda-calculus and the capabilities of pattern matching with hedge variables, i.e., variables that can be instantiated by any finite sequence of terms. We propose a generic confluence proof, where the way pattern abstractions are applied in a non-deterministic calculus is axiomatized. Moreover, we present a typed version of the calculus and show that it satisfies the subject reduction and strong normalization properties.

1 Introduction

In the foreword to [19], Bernhard Steffen characterized the pattern calculus as an attempt "to provide a unifying approach, bridging the gaps between different programming styles and paradigms according to a new slogan – computation is pattern matching." Patterns describe data structures. Matching provides functionality. The more flexible the patterns are, the more powerful the calculus is. Dynamic patterns are the most expressive ones: they can be instantiated, generated, and reduced.

There have been several approaches to design pattern calculi, extending the λ-calculus with pattern matching. One can mention the $\lambda\phi$-calculus [35], Pure Pattern Calculus [20], the ρ-calculus [10], the λ–calculus with constructors [2], and the typed versions thereof, e.g., [5, 30].

Pattern calculi are expressive, but, on the other hand, there is a price to pay for that: Confluence is lost and various restrictions have to be imposed to recover it. Cirstea and Faure in [9] proposed a generic confluence proof for dynamic pattern calculi with unitary matching algorithm. There are some conditions the matching function should satisfy, in order the guarantee the confluence.

Strong normalization is another property which does not come granted. For instance, for quite some time it was an open problem whether the Pure Pattern Calculus is strongly normalizing, until the positive answer was given in [38].

In this paper, we introduce yet another dynamic pattern calculus, which permits the use of hedge variables. These are variables which can be instantiated by finite, possibly empty, sequences of terms, called hedges. Pattern matching with hedge variables has interesting applications in programming languages [40, 39,

7], rewriting [18], knowledge representation [27], logic [24, 21], program synthesis [8], XML processing [11]. An example of such a pattern is $f(X, a, Y)$, where X and Y are hedge variables. It matches, e.g., a term $f(a, b, a, c)$ in two different ways with $\{X \mapsto \epsilon, Y \mapsto \langle b, a, c\rangle\}$ and $\{X \mapsto \langle a, b\rangle, Y \mapsto c\}$, where ϵ is the empty sequence. Our calculus permits hedge variables in the argument positions.

Matching with hedge variables is finitary, i.e., may have finitely many incomparable solutions, thus applying a function with a pattern-abstraction gives rise to different possible results. This makes the resulting calculus non-deterministic, but rather than having a purely non-deterministic notion of evaluation, we define a reduction which introduces a formal sum of terms, representing all possible results of a non-deterministic computation. We generalize the calculus to a general formalism for pattern-based calculi with finitary matching and give sufficient confluence conditions for the general case.

After having introduced the untyped calculus we present a typed version of it, which essentially extends simple types by star types and introduces a corresponding subtyping relation. We then show the subject reduction property of our type system, i.e., that evaluation preserves types, and the termination of the evaluation of well-typed terms (strong normalization of the typed calculus). Technically our approach is similar to the proofs of similar properties for λ-calculus with the extra difficulties of ensuring the correct handling of the matching process and the non-determinism of the calculus.

Compared to the other dynamic pattern calculi, the main contributions of this paper are dealing with hedge variables, formulating the generic confluence proof for finitary matching, and proving subject reduction and strong normalization properties for the typed version of the calculus with hedge variables. The other related work is briefly discussed below.

Languages with hedge variables and unranked terms: There are programming languages that deal with hedge variables. Probably the most prominent one is Mathematica [40], with a powerful rule-based programming language that uses (essentially first order, equational) matching with hedge variables [23]. Systems such as Pρlog [14] and XCentric [12] extend logic programming with the capabilities of solving equations over hedge variables. Pρlog is based on rewriting, it has an underlying operational semantics based on SLD resolution with backtracking (See also Sect. 4). XCentric is a first-order logic programming language with an SLD-resolution based semantics. The main difference from our formalism is that the calculus presented here is functional, higher-order and, along the lines of the λ-calculus, is both a general computational model and a foundation for a functional programming language with hedge variables.

Regular type systems: Regular type systems were defined before, mostly motivated by XML-processing. Examples are the type systems of XDuce [16], CDuce [6], and XHaskell [34]. These systems have regular expression types that play a key role in the pattern matching algorithm itself, i.e., pattern matching is done with types, not between values. They use semantic subtyping as the underlying theory and are not based on an algebra of unranked trees – the unranked properties of these systems are obtained indirectly by typing with regular types.

Our calculus is based directly on the elegant algebraic theory of unranked terms and hedge variables. It is closely related to the λ-calculus, in the sense that all functions are curried, thus we have a natural way of expressing partial application of a function. Our language also has a clear separation between terms and types: pattern-matching is defined in the term language and types are used only to filter values and guarantee correctness with respect to the type system.

Non-determinism: The idea of transforming a non-deterministic calculus into a deterministic one with the help of sums is not new. It has been exploited in a way or another in, e.g., differential λ-calculus [15], linear-algebraic λ-calculus [3, 13], resource calculus [29]. The idea closest to us is described in [13], as a non-deterministic extension of the call-by-value λ-calculus, which corresponds to the additive fragment of the linear-algebraic λ-calculus. The $+$ there is associative, commutative, and distributes over application both from right (corresponds to parallel composition) and left (corresponds to call-by-value). There is also the neutral element for the sum, expressing the impossible computation. The authors define a type system and prove subject reduction and strong normalization.

In our calculus, sums originate from sets of different matchers. Hence, besides being associative and commutative, $+$ is also idempotent. We do not have the neutral element: This would correspond to the case when there is no matcher between a pattern and a term. But in this case the reduction is not possible. We do not introduce a term to express this impossible reduction in the language itself. Like the sum in [13], our $+$ is also left and right distributive over applications. Moreover, our language has the other features as well: Patterns, unranked symbols, and hedge variables, which really make a difference.

2 The Untyped Calculus

In this section, we define the syntax and the operational semantics of the untyped dynamic pattern calculus with hedge variables and study its confluence.

2.1 Syntax

The alphabet consists of the set $\mathcal{X}_{\mathrm{t}}$ of term variables, $\mathcal{X}_{\mathrm{h}}$ of hedge variables and $\mathcal{F}$ of constants. They are disjoint and countably infinite. The symbols x, X and f range over $\mathcal{X}_{\mathrm{t}}, \mathcal{X}_{\mathrm{h}}$ and $\mathcal{F}$, respectively, and v ranges over the set $\mathcal{X}_{\mathrm{t}} \cup \mathcal{X}_{\mathrm{h}}$.

Terms are defined by the following grammar:

$$M, N ::= x \mid f \mid (M\,N) \mid (M\,X) \mid \lambda_{\mathcal{V}} M.N \mid M + N$$

where $(M\,N)$ is an *application of a term to a term* and $(M\,X)$ is an *application of a term to a hedge variable.* The set $\mathcal{V}$ in the *abstraction* $\lambda_{\mathcal{V}} M.N$ is a subset of the set of variables of M and represents the set of variables bound by the abstraction. In $\lambda_{\mathcal{V}} M.N$ we call the term M a *pattern.* We consider $+$ to be associative, commutative, and idempotent that distributes over applications. We write ACID for this property. Distributivity is both left and right distributivity.

The letters M, N, P, Q, W are used to denote terms, while s is used for a hedge variable or a term. Application associates to the left, therefore we can write $(M\, s_1\, \cdots\, s_n)$ for $((\cdots(M\, s_1)\cdots)s_n)$. When there is no ambiguity, the outermost parentheses are omitted as well. The set of terms is denoted by $\mathcal{T}$.

The *sets of free and bound variables* of a term P, denoted $\mathtt{fv}(P)$ and $\mathtt{bv}(P)$ respectively, are defined inductively as follows:

$$\begin{array}{ll}
\mathtt{fv}(x) = \{x\} \qquad \mathtt{fv}(f) = \varnothing & \mathtt{bv}(x) = \varnothing \qquad \mathtt{bv}(f) = \varnothing \\
\mathtt{fv}(M\,N) = \mathtt{fv}(M) \cup \mathtt{fv}(N) & \mathtt{bv}(M\,N) = \mathtt{bv}(M) \cup \mathtt{bv}(N) \\
\mathtt{fv}(M\,X) = \mathtt{fv}(M) \cup \{X\} & \mathtt{bv}(M\,X) = \mathtt{bv}(M) \\
\mathtt{fv}(\lambda_{\mathcal{V}} M.N)) = (\mathtt{fv}(M) \cup \mathtt{fv}(N)) \backslash \mathcal{V} & \mathtt{bv}(\lambda_{\mathcal{V}} M.N) = \mathtt{bv}(M) \cup \mathtt{bv}(N) \cup \mathcal{V} \\
\mathtt{fv}(M + N) = \mathtt{fv}(M) \cup \mathtt{fv}(N) & \mathtt{bv}(M + N) = \mathtt{bv}(M) \cup \mathtt{bv}(N)
\end{array}$$

Note that, unlike the λ-calculus, we abstract not only on variables but on terms. The set of variables bound by an abstraction is not necessarily the same as the set of free variables of the corresponding pattern. Intuitively, in the term $\lambda_{\mathcal{V}} M.N$, the set $\mathcal{V}$ represents the set of variables bound by the abstraction and $\mathcal{V} \subseteq \mathtt{fv}(M)$. We adopt Barendregt's variable name convention [4], i.e. free and bound variables have different names. This can be fulfilled by renaming bound variables. We identify terms modulo α-equivalence, therefore expressions that differ only in the names of bound variables are identified.

Below we assume that terms are in the ACID normal form with respect to $+$ and application. First-order ACID-unification and matching are decidable and the rewrite system corresponding to the ACID equations is convergent [1].

Example 1. The term $\lambda_{\{x,X\}}(fx + gx)X.fx(gx + gX + gx)$ is not in the ACID normal form, while $\lambda_{\{x,X\}}(fxX + gxX).(fx(gx) + fx(gX))$ is.

Note that our functions do not take multiple arguments (or an n-tuple of arguments). They are defined in their Curry form, i.e. a function with multiple arguments is represented as a chain of functions each with a single argument.

Hedges are finite (possible empty) sequences of terms and hedge variables. We use h to denote them. For the empty hedge we use ϵ. For readability, we put hedges in angle brackets if they have more than one element, e.g., $\langle M, X, N\rangle$.

A *substitution* is a mapping from term variables to terms, and from hedge variables to hedges, such that all but finitely many term and hedge variables are mapped to themselves. We use φ and ϑ to denote substitutions. Each substitution φ is represented as a finite set of pairs $\{v_1 \mapsto \varphi(v_1), \ldots, v_n \mapsto \varphi(v_n)\}$ where the v's are all those variables for which $\varphi(v_i) \neq v_i$. The sets $\mathit{Dom}(\varphi) = \{v_1, \ldots, v_n\}$ and $\mathit{Ran}(\varphi) = \{\varphi(v_1), \ldots, \varphi(v_n)\}$ are called the *domain* and the *range* of φ, respectively. The set $\mathit{Var}(\phi)$ is defined as $\mathit{Var}(\phi) = \mathit{Dom}(\varphi) \cup \mathtt{fv}(\mathit{Ran}(\varphi))$.

The *application* of a substitution φ to a term M replaces each *free* occurrence of a variable v in M with $\varphi(v)$. It is defined inductively:

$$\begin{array}{ll}
x\varphi = \varphi(x), \text{ if } x \in \mathit{Dom}(\varphi). & (M\,X)\varphi = M\varphi\, s_1, \ldots s_n, \\
x\varphi = x, \text{ if } x \notin \mathit{Dom}(\varphi). & \text{if } \varphi(X) = \langle s_1, \ldots, s_n\rangle.
\end{array}$$

$$f\varphi = f. \qquad (M\,X)\varphi = M\varphi \text{ if } \varphi(X) = \epsilon.$$
$$(M\,N)\varphi = M\varphi\,N\varphi. \qquad (M\,Y)\varphi = M\varphi\,Y \text{ if } Y \notin \mathit{Dom}(\varphi).$$
$$(M + N)\varphi = M\varphi + N\varphi. \qquad (\lambda_{\mathcal{V}} M.N)\varphi = \lambda_{\mathcal{V}} M\varphi.N\varphi.$$

In the abstraction, it is assumed that $\mathit{Var}(\varphi) \cap \mathcal{V} = \varnothing$. This can be achieved by properly renaming the bound variables. Hence, the equality here is α-equivalence.

The remark above about the Curry form explains how substitution is defined in our calculus, especially the rule for substituting a hedge variable X by a hedge $\langle s_1, \ldots, s_n \rangle$ in a term MX, that, to be consistent with the Curry form of terms, gives the resulting term $(\cdots (M\varphi\, s_1) \cdots s_n)$, written as $M\varphi\, s_1, \ldots s_n$.

Example 2. Let $M = \lambda_{\{x,Y\}} fXxY.\, y(gX)xZ$ and $\varphi = \{x \mapsto gx, y \mapsto \lambda_x x.fxa, Z \mapsto \epsilon, X \mapsto \langle \lambda_x x.x, \lambda_x x.(x + fx) \rangle\}$. Then $M\varphi$ (in the ACID normal form) is

$$M\varphi = \lambda_{\{x',Y\}} (f(\lambda_x x.x)(\lambda_x x.x)x'Y + f(\lambda_x x.x)(\lambda_x x.fx)x'Y)\,.$$
$$(\lambda_x x.fxa)(g(\lambda_x x.x)(\lambda_x x.x))x' + (\lambda_x x.fxa)(g(\lambda_x x.x)(\lambda_x x.fx))x'.$$

The *composition* of substitutions is defined in the standard way, as the composition of two mappings. We use juxtaposition to denote it, writing $\varphi\vartheta$ for the composition of φ and ϑ. For all M, we have $M\varphi\vartheta = (M\varphi)\vartheta$.

The *restriction* of a substitution φ to a set of variables V, denoted $\varphi|_V$, is defined as $\varphi|_V(v) = \varphi(v)$ if $v \in V$, and $\varphi|_V(v) = v$ otherwise.

2.2 Reduction

Evaluation in the dynamic pattern calculus with hedge variables is given by three binary relations β_1, β_2, and β_3 on terms, written in the form of reduction rules below. The relation β_1 defines the way how pattern-abstractions are applied. It is parametrized by a function (the pattern matching algorithm) Sol, which takes as parameters two terms M and Q and a set of variables $\mathcal{V}$ and returns a finite set of substitutions. We denote it by $\mathit{Sol}(M \lessdot_{\mathcal{V}} Q)$. The relations β_2 and β_3 define how + distributes over abstraction:

$\beta_1 : (\lambda_{\mathcal{V}} M.N)\,Q \to N\varphi_1 + \cdots + N\varphi_n$, where M and Q are not of the form $W_1 + W_2$ and $\mathit{Sol}(M \lessdot_{\mathcal{V}} Q) = \{\varphi_1, \ldots, \varphi_n\},\ n \geqslant 1$.

$\beta_2 : \lambda_{\mathcal{V}} M_1 + M_2.N \to \lambda_{\mathcal{V}} M_1.N + \lambda_{\mathcal{V}} M_2.N$.

$\beta_3 : \lambda_{\mathcal{V}} M.N_1 + N_2 \to \lambda_{\mathcal{V}} M.N_1 + \lambda_{\mathcal{V}} M.N_2$.

A binary relation of compatibility $\to_R$ is defined with the help of the inference rules below. They hold for any M, N, M', N', X, and $\mathcal{V}$.

$$\frac{M \to_R M'}{NM \to_R NM'} \qquad \frac{M \to_R M'}{MN \to_R M'N} \qquad \frac{M \to_R M'}{MX \to_R M'X}$$

$$\frac{M \to_R M'}{M + N \to_R M' + N} \qquad \frac{M \to_R M'}{\lambda_{\mathcal{V}} M.N \to_R \lambda_{\mathcal{V}} M'.N} \qquad \frac{N \to_R N'}{\lambda_{\mathcal{V}} M.N \to_R \lambda_{\mathcal{V}} M.N'}$$

In what follows, $\to_{\beta_C}$ denotes the compatible closure of the union of β_1, β_2 and β_3 relations and $\twoheadrightarrow_{\beta_C}$ denotes the reflexive and transitive closure of $\to_{\beta_C}$.

Example 3 (Matching with Hedge Variables). As a concrete example, consider matching between terms with hedge variables without λ-abstractions and $+$. A matching equation is written as $P \ll^? M$. Its solution (a matcher) is a substitution φ such that $P\varphi = M$. Matching problem Π is a finite set of matching equations. A state is a pair $\langle \Pi, \vartheta \rangle$ of a matching problem and a substitution. The matching procedure [22, 26] with hedge variables is reformulated here as a set of state transformations of the form $\langle \{P \ll^? M\} \cup \Pi, \vartheta \rangle \rightsquigarrow \langle (\Pi \cup \Pi')\varphi, \vartheta\varphi \rangle$. Each of the transformations is characterized by a rule of the form $P \ll^? M \rightsquigarrow_\varphi \Pi'$. There are four rules: (ε stands for the identity substitution.)

$$M \ll^? M \rightsquigarrow_\varepsilon \varnothing.$$
$$P_1\, P_2 \ll^? M_1\, M_2 \rightsquigarrow_\varepsilon \{P_1 \ll^? M_1, P_2 \ll^? M_2\}, \text{where } P_1\, P_2 \neq M_1\, M_2.$$
$$x \ll^? M \rightsquigarrow_{\{x \mapsto M\}} \varnothing.$$
$$P\, X \ll^? M s_1 \cdots s_n\, s'_1 \cdots s'_m \rightsquigarrow_{\{X \mapsto \langle s'_1, \ldots, s'_m \rangle\}} \{P \ll^? M s_1 \cdots s_n\},\ n, m \geqslant 0.$$

We can define $Sol(P \lessdot_{\mathcal{V}} M)$ as a partial function from the triple $P, M, \mathcal{V}$ (where $\mathtt{fv}(M) \cap \mathcal{V} = \varnothing$) to a set of λ-abstraction- and $+$-free substitutions with the following conditions:

C1. If P or M contains a λ-abstraction or $+$, or if $\mathtt{fv}(P) \neq \mathcal{V}$, then *Sol* is undefined.
C2. Otherwise, $Sol(P \lessdot_{\mathcal{V}} M)$ transforms $P \ll^? M$ by the rules above in all possible ways as long as possible and collects substitutions φ from the success states $\langle \varnothing, \varphi \rangle$ in the set $\mathcal{M}$ of computed matchers (which is complete and finite [26]). If $\mathcal{M}$ is empty, then $Sol(P \lessdot_{\mathcal{V}} M)$ is undefined. Otherwise $Sol(P \lessdot_{\mathcal{V}} M) = \mathcal{M}$.

Then the following example illustrates $\rightarrow_{\beta_C}$ reductions with the just defined *Sol*: $(\lambda_{\{X,Y\}} fXY.\, \lambda_{\{x\}} x.gXx)fa \rightarrow_{\beta_C} \lambda_{\{x\}} x.(gx + gax) \rightarrow_{\beta_C} \lambda_{\{x\}} x.gx + \lambda_{\{x\}} x.gax$.

2.3 Confluence

The function *Sol* defined in Example 3 above leads to diverging reductions:

Example 4. Let *Sol* be the function based on the matching with hedge variables as defined in 3. Then the term $M = (\lambda_{\{Z\}} fZ.(\lambda_{\{X,Y\}} fXY.fX)fZ)fab$ reduces to two different normal forms:

1. $M \rightarrow_{\beta_C} (\lambda_Z fZ.(f + fZ))fab \rightarrow_{\beta_C} f + fab$.
2. $M \rightarrow_{\beta_C} (\lambda_{\{X,Y\}} fXY.fX)fab \rightarrow_{\beta_C} f + fa + fab$.

Our goal is to impose restrictions on *Sol* so that confluence is guaranteed. The confluence proof will be based on the standard method due to Tait and Martin-Löf [4]. It requires the notion of *parallel reduction* which is defined as follows:

$$\frac{}{s \Rightarrow_{\beta_C} s} \qquad \frac{s_1 \Rightarrow_{\beta_C} s'_1 \quad \ldots \quad s_n \Rightarrow_{\beta_C} s'_n}{\langle s_1, \ldots, s_n \rangle \Rightarrow_{\beta_C} \langle s'_1, \ldots, s'_n \rangle} \qquad \frac{M \Rightarrow_{\beta_C} M' \quad s \Rightarrow_{\beta_C} s'}{M\, s \Rightarrow_{\beta_C} M'\, s'}$$

$$\frac{M \Rightarrow_{\beta_C} M' \quad N \Rightarrow_{\beta_C} N'}{\lambda_{\mathcal{V}} M.N \Rightarrow_{\beta_C} \lambda_{\mathcal{V}} M'.N'} \qquad \frac{M_1 \Rightarrow_{\beta_C} M_1' \quad M_2 \Rightarrow_{\beta_C} M_2' \quad N \Rightarrow_{\beta_C} N'}{\lambda_{\mathcal{V}}(M_1 + M_2).N \Rightarrow_{\beta_C} \lambda_{\mathcal{V}} M_1'.N' + \lambda_{\mathcal{V}} M_2'.N'}$$

$$\frac{M \Rightarrow_{\beta_C} M' \quad N \Rightarrow_{\beta_C} N'}{M + N \Rightarrow_{\beta_C} M' + N'} \qquad \frac{M \Rightarrow_{\beta_C} M' \quad N_1 \Rightarrow_{\beta_C} N_1' \quad N_2 \Rightarrow_{\beta_C} N_2'}{\lambda_{\mathcal{V}} M.(N_1 + N_2) \Rightarrow_{\beta_C} \lambda_{\mathcal{V}} M'.N_1' + \lambda_{\mathcal{V}} M'.N_2'}$$

$$\frac{M \Rightarrow_{\beta_C} M' \quad N \Rightarrow_{\beta_C} N' \quad Q \Rightarrow_{\beta_C} Q'}{(\lambda_{\mathcal{V}} M.N)Q \Rightarrow_{\beta_C} N'\varphi_1 + \cdots + N'\varphi_n}, \text{ if } Sol(M' \lessdot_{\mathcal{V}} Q') = \{\varphi_i \mid 1 \leqslant i \leqslant n\}.$$

Note that the parallel reduction is compatible. Its definition is extended to substitutions having the same domain by setting $\varphi \Rightarrow_{\beta_C} \varphi'$ if for all $v \in Dom(\varphi) = Dom(\varphi')$, we have $v\varphi \Rightarrow_{\beta_C} v\varphi'$.

Now we are ready to define the conditions imposed on Sol. As already mentioned, the terms are in the ACID normal form.

$\mathbf{H}_0$: If $\varphi \in Sol(P \lessdot_{\mathcal{V}} M)$, then $Dom(\varphi) = \mathcal{V}$ and $\mathtt{fv}(Ran(\varphi)) \subseteq \mathtt{fv}(M)$.
$\mathbf{H}_1$: If $Sol(P \lessdot_{\mathcal{V}} M) = \{\varphi_1, \ldots, \varphi_n\}$, $n \geqslant 1$, then for all ϑ with $Var(\vartheta) \cap \mathcal{V} = \varnothing$, we have $Sol(P\vartheta \lessdot_{\mathcal{V}} M\vartheta) = \{(\varphi_1\vartheta)|_{\mathcal{V}}, \ldots, (\varphi_n\vartheta)|_{\mathcal{V}}\}$.
$\mathbf{H}_2$: If $Sol(P \lessdot_{\mathcal{V}} M) = \{\varphi_1, \ldots, \varphi_n\}$, $n \geqslant 1$, $P \Rightarrow_{\beta_C} P'$, and $M \Rightarrow_{\beta_C} M'$, then $Sol(P' \lessdot_{\mathcal{V}} M') = \{\varphi_1', \ldots, \varphi_m'\}$, $m \geqslant 1$, and
(a) for all $1 \leqslant i \leqslant n$ there exists $1 \leqslant j \leqslant m$ such that $\varphi_i \Rightarrow_{\beta_C} \varphi_j'$ and
(b) for all $1 \leqslant j \leqslant m$ there exists $1 \leqslant i \leqslant n$ such that $\varphi_i \Rightarrow_{\beta_C} \varphi_j'$.

These conditions extend the ones for the core dynamic pattern calculus from [9]. We will show that they are sufficient for proving confluence of our calculus. We assume that the relations $\rightarrow_{\beta_C}$ and $\Rightarrow_{\beta_C}$ in the lemmas and in the theorem below use a Sol which satisfies $\mathbf{H}_0$, $\mathbf{H}_1$, and $\mathbf{H}_2$. We do not state this explicitly in the conditions.

Lemma 1. *The following inclusion hold:* $\rightarrow_{\beta_C} \subseteq \Rightarrow_{\beta_C} \subseteq \twoheadrightarrow_{\beta_C}$.

Proof. First we prove $\rightarrow_{\beta_C} \subseteq \Rightarrow_{\beta_C}$.

Assume that $M = (\lambda_{\mathcal{V}} M_1.M_2)M_3$ reduces by β_1 to $N = \sum_{i=1}^{n} M_2\varphi_i$ where $Sol(M_1 \lessdot_{\mathcal{V}} M_3) = \{\varphi_1, \ldots, \varphi_n\}$. Then we trivially have $M \Rightarrow_{\beta_C} N$, since the relation $\Rightarrow_{\beta_C}$ is reflexive. If $M = \lambda_{\mathcal{V}} M_1 + M_2.M_3$ reduces to $N = \lambda_{\mathcal{V}} M_1.M_3 + \lambda_{\mathcal{V}} M_2.M_3$ by β_2 or $M = \lambda_{\mathcal{V}} M_1.M_2 + M_3$ reduces to $N = \lambda_{\mathcal{V}} M_1.M_2 + \lambda_{\mathcal{V}} M_1.M_3$ by β_3 then again by reflexivity of $\Rightarrow_{\beta_C}$we have $M \Rightarrow_{\beta_C} N$. In the other cases (the reduction does not occur at the head position), the proof directly follows from the compatibility of the relation $\Rightarrow_{\beta_C}$.

Now we prove by induction that $M \Rightarrow_{\beta_C} N$ implies $M \twoheadrightarrow_{\beta_C} N$.

- If $M = N$ then the result follows from reflexivity of $\twoheadrightarrow_{\beta_C}$.
- If $M = M_1X \Rightarrow_{\beta_C} N = N_1X$ with $M_1 \Rightarrow_{\beta_C} N_1$, then by the induction hypothesis applied to M_1 we have $M_1 \twoheadrightarrow_{\beta_C} N_1$. Since $\twoheadrightarrow_{\beta_C}$ is compatible, we get $M_1X \twoheadrightarrow_{\beta_C} N_1X$.
- If $M = M_1M_2 \Rightarrow_{\beta_C} N = N_1N_2$ with $M_1 \Rightarrow_{\beta_C} N_1$ and $M_2 \Rightarrow_{\beta_C} N_2$, then by the induction hypothesis applied to M_1 and M_2 we have $M_1 \twoheadrightarrow_{\beta_C} N_1$ and $M_2 \twoheadrightarrow_{\beta_C} N_2$. Since $\twoheadrightarrow_{\beta_C}$ is compatible, we have $M_1M_2 \twoheadrightarrow_{\beta_C} N_1M_2$ and $N_1M_2 \twoheadrightarrow_{\beta_C} N_1N_2$, and by transitivity of $\twoheadrightarrow_{\beta_C}$ we get $M_1M_2 \twoheadrightarrow_{\beta_C} N_1N_2$.

- If $M = \lambda_{\mathcal{V}} M_1.M_2 \Rightarrow_{\beta_C} N = \lambda_{\mathcal{V}} N_1.N_2$ with $M_1 \Rightarrow_{\beta_C} N_1$ and $M_2 \Rightarrow_{\beta_C} N_2$. Similar to the previous case.
- If $M = M_1 + M_2 \Rightarrow_{\beta_C} N = N_1 + N_2$ with $M_1 \Rightarrow_{\beta_C} N_1$ and $M_2 \Rightarrow_{\beta_C} N_2$, then by the induction hypothesis applied to M_1 and M_2 we have $M_1 \twoheadrightarrow_{\beta_C} N_1$ and $M_2 \twoheadrightarrow_{\beta_C} N_2$. Since $\twoheadrightarrow_{\beta_C}$ is compatible, we have $M_1 + M_2 \twoheadrightarrow_{\beta_C} N_1 + M_2$ and $M_2 + N_1 \twoheadrightarrow_{\beta_C} N_2 + N_1$. Since $+$ is ACID, we can write $M_2 + N_1 \twoheadrightarrow_{\beta_C} N_2 + N_1$ as $N_1 + M_2 \twoheadrightarrow_{\beta_C} N_1 + N_2$ and by the transitivity of $\twoheadrightarrow_{\beta_C}$ we get $M_1 + M_2 \twoheadrightarrow_{\beta_C} N_1 + N_2$.
- If $M = (\lambda_{\mathcal{V}} M_1.M_2)M_3 \Rightarrow_{\beta_C} N = \sum_{i=1}^{n} N_2\varphi_i$ with $M_1 \Rightarrow_{\beta_C} N_1, M_2 \Rightarrow_{\beta_C} N_2, M_3 \Rightarrow_{\beta_C} N_3$ and $Sol(N_1 \ll_{\mathcal{V}} N_3) = \{\varphi_1, \ldots, \varphi_n\}$, then by the induction hypothesis we have $M_1 \twoheadrightarrow_{\beta_C} N_1, M_2 \twoheadrightarrow_{\beta_C} N_2, M_3 \twoheadrightarrow_{\beta_C} N_3$. By compatibility and transitivity of $\twoheadrightarrow_{\beta_C}$ we have, first, $\lambda_{\mathcal{V}} M_1.M_2 \twoheadrightarrow_{\beta_C} \lambda_{\mathcal{V}} N_1.N_2$ and then $(\lambda_{\mathcal{V}} M_1.M_2)M_3 \twoheadrightarrow_{\beta_C} (\lambda_{\mathcal{V}} N_1.N_2)N_3$. By β_1 we have $(\lambda_{\mathcal{V}} N_1.N_2)N_3 \rightarrow_{\beta_C} \sum_{i=1}^{n} N_2\varphi_i$. Hence, $(\lambda_{\mathcal{V}} M_1.M_2)M_3 \twoheadrightarrow_{\beta_C} \sum_{i=1}^{n} N_2\varphi_i$.
- If $M = \lambda_{\mathcal{V}} M_1 + M_2.M_3 \Rightarrow_{\beta_C} N = \lambda_{\mathcal{V}} N_1.N_3 + \lambda_{\mathcal{V}} N_2.N_3$ with $M_1 \Rightarrow_{\beta_C} N_1, M_2 \Rightarrow_{\beta_C} N_2$ and $M_3 \Rightarrow_{\beta_C} N_3$ then by induction hypothesis we have $M_1 \twoheadrightarrow_{\beta_C} N_1, M_2 \twoheadrightarrow_{\beta_C} N_2$ and $M_3 \twoheadrightarrow_{\beta_C} N_3$. By compatibility and transitivity of $\twoheadrightarrow_{\beta_C}$ we have $\lambda_{\mathcal{V}} M_1 + M_2.M_3 \twoheadrightarrow_{\beta_C} \lambda_{\mathcal{V}} N_1 + N_2.N_3$. By β_2 we have $\lambda_{\mathcal{V}} N_1 + N_2.N_3 \rightarrow_{\beta_C} \lambda_{\mathcal{V}} N_1.N_3 + \lambda_{\mathcal{V}} N_2.N_3$ therefore we get $\lambda_{\mathcal{V}} M_1 + M_2.M_3 \twoheadrightarrow_{\beta_C} \lambda_{\mathcal{V}} N_1.N_3 + \lambda_{\mathcal{V}} N_2.N_3$.
- If $M = \lambda_{\mathcal{V}} M_1.M_2 + M_3 \Rightarrow_{\beta_C} N = \lambda_{\mathcal{V}} N_1.N_2 + \lambda_{\mathcal{V}} N_1.N_3$ with $M_1 \Rightarrow_{\beta_C} N_1, M_2 \Rightarrow_{\beta_C} N_2$ and $M_3 \Rightarrow_{\beta_C} N_3$. Similar to the previous case.

□

Lemma 2 (Fundamental Lemma). *For all terms M and M' and for all substitutions φ and φ' with $Dom(\varphi) = Dom(\varphi')$, if $M \Rightarrow_{\beta_C} M'$ and $\varphi \Rightarrow_{\beta_C} \varphi'$, then $M\varphi \Rightarrow_{\beta_C} M'\varphi'$.*

Proof. By induction on the $\Rightarrow_{\beta_C}$ relation.

- If $M = M'$, then the proof follows by induction on M.
  - If $M = x$ and $x \in Dom(\varphi)$, then by the definition of parallel reduction for substitutions we have $x\varphi \Rightarrow_{\beta_C} x\varphi'$.
  - If $M = y$ and $y \notin Dom(\varphi)$, then by the assumption we have $y \notin Dom(\varphi')$. By the definition of substitution application, since $y\varphi = y\varphi' = y$, we can conclude $y\varphi \Rightarrow_{\beta_C} y\varphi'$.
  - If $M = f$, then by the definition substitution application, we have $f\varphi = f$ and $f\varphi' = f$. By the definition of parallel reduction, we get $f\varphi \Rightarrow_{\beta_C} f\varphi'$.
  - If $M = M_1 X$ with $M_1 \Rightarrow_{\beta_C} M_1$, then by the induction hypothesis we have $M_1\varphi \Rightarrow_{\beta_C} M_1\varphi'$. We consider two cases:
    - If $X \notin Dom(\varphi)$, then by hypothesis we have $X \notin Dom(\varphi')$. Therefore, by the definitions of substitution application and parallel reduction, $(M_1X)\varphi = (M_1\varphi)X \Rightarrow_{\beta_C} (M_1\varphi')X = (M_1X)\varphi'$.
    - If $X \in Dom(\varphi)$ then by the definition of parallel reduction for substitutions, we have $X\varphi \Rightarrow_{\beta_C} X\varphi'$. We have two cases: either $\varphi(X) = \epsilon$ in which case $\varphi'(X) = \epsilon$, or $\varphi(X) = (s_1, \ldots, s_n)$ in which case

$\varphi'(X) = (s'_1, \ldots, s'_n)$, with $s_1 \Rightarrow_{\beta_C} s'_1, \ldots, s_n \Rightarrow_{\beta_C} s'_n$. By the definitions of parallel reduction and substitution application, we have in the first case $(M_1 X)\varphi = M_1\varphi \Rightarrow_{\beta_C} M_1\varphi' = (M_1 X)\varphi'$, and in the second case $(M_1 X)\varphi = (\cdots((M_1\varphi)s_1)\cdots s_n) \Rightarrow^*_{\beta_C} (\cdots((M_1\varphi')s'_1)\cdots s'_n) = (M_1 X)\varphi'$.

- If $M = M_1 M_2$ with $M_1 \Rightarrow_{\beta_C} M_1$ and $M_2 \Rightarrow_{\beta_C} M_2$ by induction hypothesis we have $M_1\varphi \Rightarrow_{\beta_C} M_1\varphi'$ and $M_2\varphi \Rightarrow_{\beta_C} M_2\varphi'$. By the definitions of parallel reduction and substitution application, we have $(M_1 M_2)\varphi \Rightarrow_{\beta_C} (M_1 M_2)\varphi'$.
- If $M = \lambda_{\mathcal{V}} M_1.M_2$ with $M_1 \Rightarrow_{\beta_C} M_1$ and $M_2 \Rightarrow_{\beta_C} M_2$, then we proceed similar to the previous case.
- If $M_1 + M_2 \Rightarrow_{\beta_C} M_1 + M_2$ with $M_1 \Rightarrow_{\beta_C} M_1$ and $M_2 \Rightarrow_{\beta_C} M_2$, then we proceed similar to the previous cases.
- If $(\lambda_{\mathcal{V}} M_1.M_2)M_3 \Rightarrow_{\beta_C} (\lambda_{\mathcal{V}} M_1.M_2)M_3$ with $M_1 \Rightarrow_{\beta_C} M_1, M_2 \Rightarrow_{\beta_C} M_2$ and $M_3 \Rightarrow_{\beta_C} M_3$, then by the induction hypothesis applied to M_1, M_2 and M_3 we have $M_1\varphi \Rightarrow_{\beta_C} M_1\varphi', M_2\varphi \Rightarrow_{\beta_C} M_2\varphi', M_3\varphi \Rightarrow_{\beta_C} M_3\varphi'$. By the definitions of parallel reduction and substitution application, we have $((\lambda_{\mathcal{V}} M_1.M_2)M_3)\varphi \Rightarrow_{\beta_C} ((\lambda_{\mathcal{V}} M_1.M_2)M_3)\varphi'$.
- If $\lambda_{\mathcal{V}} M_1 + M_2.M_3 \Rightarrow_{\beta_C} \lambda_{\mathcal{V}} M_1 + M_2.M_3$ with $M_1 \Rightarrow_{\beta_C} M_1, M_2 \Rightarrow_{\beta_C} M_2$ and $M_3 \Rightarrow_{\beta_C} M_3$, then by the induction hypothesis applied to M_1, M_2 and M_3 we have $M_1\varphi \Rightarrow_{\beta_C} M_1\varphi', M_2\varphi \Rightarrow_{\beta_C} M_2\varphi', M_3\varphi \Rightarrow_{\beta_C} M_3\varphi'$. By the definitions of parallel reduction and substitution application, we have $(\lambda_{\mathcal{V}} M_1 + M_2.M_3)\varphi \Rightarrow_{\beta_C} (\lambda_{\mathcal{V}} M_1 + M_2.M_3)\varphi'$.
- If $\lambda_{\mathcal{V}} M_1.M_2 + M_3 \Rightarrow_{\beta_C} \lambda_{\mathcal{V}} M_1.M_2 + M_3$ with $M_1 \Rightarrow_{\beta_C} M_1, M_2 \Rightarrow_{\beta_C} M_2$ and $M_3 \Rightarrow_{\beta_C} M_3$, then by the induction hypothesis applied to M_1, M_2 and M_3 we have $M_1\varphi \Rightarrow_{\beta_C} M_1\varphi', M_2\varphi \Rightarrow_{\beta_C} M_2\varphi', M_3\varphi \Rightarrow_{\beta_C} M_3\varphi'$. By the definitions of parallel reduction and substitution application, we have $(\lambda_{\mathcal{V}} M_1.M_2 + M_3)\varphi \Rightarrow_{\beta_C} (\lambda_{\mathcal{V}} M_1.M_2 + M_3)\varphi'$.

– If $M = M_1 X \Rightarrow_{\beta_C} M' = M'_1 X$ with $M_1 \Rightarrow_{\beta_C} M'_1$, then by the induction hypothesis we have $M_1\varphi \Rightarrow_{\beta_C} M'_1\varphi'$. We consider two cases:
  - If $X \notin \mathit{Dom}(\varphi)$, then by hypothesis we have $X \notin \mathit{Dom}(\varphi')$. Therefore, by the definitions of parallel reduction and substitution application, $(M_1 X)\varphi = (M_1\varphi)X \Rightarrow_{\beta_C} (M'_1\varphi')X = (M'_1 X)\varphi'$.
  - If $X \in \mathit{Dom}(\varphi)$ then by the definition of parallel reduction for substitutions, we have $X\varphi \Rightarrow_{\beta_C} X\varphi'$. We have two cases: either $\varphi(X) = \epsilon$ in which case $\varphi'(X) = \epsilon$, or $\varphi(X) = (s_1, \ldots, s_n)$ in which case $\varphi'(X) = (s'_1, \ldots, s'_n)$, with $s_1 \Rightarrow_{\beta_C} s'_1, \ldots, s_n \Rightarrow_{\beta_C} s'_n$. By the definitions of parallel reduction and substitution application, we have in the first case $(M_1 X)\varphi = M_1\varphi \Rightarrow_{\beta_C} M'_1\varphi' = (M'_1 X)\varphi'$, and get in the second case $(M_1 X)\varphi = (\cdots((M_1\varphi)s_1)\cdots s_n) \Rightarrow^*_{\beta_C} (\cdots((M'_1\varphi')s'_1)\cdots s'_n) = (M_1 X)\varphi'$.

– If $M = M_1 M_2 \Rightarrow_{\beta_C} M' = M'_1 M'_2$ with $M_1 \Rightarrow_{\beta_C} M'_1$ and $M_2 \Rightarrow_{\beta_C} M'_2$, then by the induction hypothesis applied to M_1 and M_2 we have $M_1\varphi \Rightarrow_{\beta_C} M'_1\varphi'$ and $M_2\varphi \Rightarrow_{\beta_C} M'_2\varphi'$. By the definitions of parallel reduction and substitution application, we have $(M_1 M_2)\varphi \Rightarrow_{\beta_C} (M'_1 M'_2)\varphi'$.

– If $M = \lambda_{\mathcal{V}} M_1.M_2 \Rightarrow_{\beta_C} M' = \lambda_{\mathcal{V}} M'_1.M'_2$ with $M_1 \Rightarrow_{\beta_C} M'_1$ and $M_2 \Rightarrow_{\beta_C} M'_2$, then we proceed similarly to the previous case.

- If $M = M_1 + M_2 \Rightarrow_{\beta_C} M' = M'_1 + M'_2$ with $M_1 \Rightarrow_{\beta_C} M'_1$ and $M_2 \Rightarrow_{\beta_C} M'_2$ then we proceed similarly to the previous cases.
- If $M = \lambda_{\mathcal{V}} M_1 + M_2.M_3 \Rightarrow_{\beta_C} M' = \lambda_{\mathcal{V}} M'_1.M'_3 + \lambda_{\mathcal{V}} M'_2.M'_3$ with $M_1 \Rightarrow_{\beta_C} M'_1, M_2 \Rightarrow_{\beta_C} M'_2$ and $M_3 \Rightarrow_{\beta_C} M'_3$ by induction hypothesis applied to M_1, M_2 and M_3 we have $M_1\varphi \Rightarrow_{\beta_C} M'_1\varphi', M_2\varphi \Rightarrow_{\beta_C} M'_2\varphi', M_3\varphi \Rightarrow_{\beta_C} M'_3\varphi'$. By the definitions of parallel reduction and substitution application we have $(\lambda_{\mathcal{V}} M_1 + M_2.M_3)\varphi \Rightarrow_{\beta_C} (\lambda_{\mathcal{V}} M'_1.M'_3 + \lambda_{\mathcal{V}} M'_2.M'_3)\varphi'$.
- If $M = \lambda_{\mathcal{V}} M_1.M_2 + M_3 \Rightarrow_{\beta_C} M' = \lambda_{\mathcal{V}} M'_1.M'_2 + \lambda_{\mathcal{V}} M'_1.M'_3$ with $M_1 \Rightarrow_{\beta_C} M'_1, M_2 \Rightarrow_{\beta_C} M'_2$ and $M_3 \Rightarrow_{\beta_C} M'_3$. Similar to the previous cases.
- If $M = (\lambda_{\mathcal{V}} M_1.M_2)M_3 \Rightarrow_{\beta_C} M' = \sum_{i=1}^{n} M'_2\tau_i$ with $M_1 \Rightarrow_{\beta_C} M'_1, M_2 \Rightarrow_{\beta_C} M'_2, M_3 \Rightarrow_{\beta_C} M'_3$ and $Sol(M'_1 \lessdot_{\mathcal{V}} M'_3) = \{\tau_1, \ldots, \tau_n\}$, then by the induction hypothesis applied to M_1, M_2 and M_3 we have $M_1\varphi \Rightarrow_{\beta_C} M'_1\varphi', M_2\varphi \Rightarrow_{\beta_C} M'_2\varphi', M_3\varphi \Rightarrow_{\beta_C} M'_3\varphi'$. By the definition of parallel reduction, we have $(\lambda_{\mathcal{V}} M_1\varphi.M_2\varphi)M_3\varphi \Rightarrow_{\beta_C} \sum_{i=1}^{k} (M'_2\varphi')\eta_i$, where $Sol(M'_1\varphi' \lessdot_{\mathcal{V}} M'_3\varphi') = \{\eta_1, \ldots, \eta_k\}$. We want to show $(\sum_{i=1}^{n} M'_2\tau_i)\varphi' = \sum_{i=1}^{k} (M'_2\varphi')\eta_i$. Since $Sol(M'_1 \lessdot_{\mathcal{V}} M'_3) = \{\tau_1, \ldots, \tau_n\}$, by $\mathbf{H}_1$ we have $Sol(M'_1\varphi' \lessdot_{\mathcal{V}} M'_3\varphi') = \{(\tau_1\varphi')_{|\mathcal{V}}, \ldots, (\tau_n\varphi')_{|\mathcal{V}}\}$ (one can always guarantee that $Var(\varphi') \cap \mathcal{V} = \varnothing$, through a change of bound variables). Since we have $Sol(M'_1\varphi' \lessdot_{\mathcal{V}} M'_3\varphi') = \{\eta_1, \ldots, \eta_k\}$, therefore, $\{(\tau_1\varphi')_{|\mathcal{V}}, \ldots, (\tau_n\varphi')_{|\mathcal{V}}\} = \{\eta_1, \ldots, \eta_k\}$ and, hence, $n = k$. So, we have to show $\tau_i\varphi' = \varphi'(\tau_i\varphi')_{|\mathcal{V}}$ holds for all $1 \leqslant i \leqslant n$, i.e., for all v, we have to show $v\tau_i\varphi' = v\varphi'(\tau_i\varphi')_{|\mathcal{V}}$. The proof proceeds as follows. The are two cases:
  - If v belongs to $\mathcal{V}$ then by $\mathbf{H}_0$ we have $v \in Dom(\tau_i)$. We want to show $v\varphi'(\tau_i\varphi')_{|\mathcal{V}} = v\tau_i\varphi'$. Since we can assume $Dom(\varphi') \cap \mathcal{V} = \varnothing$, by the definition of substitution application, we have $v\varphi'(\tau_i\varphi')_{|\mathcal{V}} = v(\tau_i\varphi')_{|\mathcal{V}}$. Since $Dom(\varphi') \cap Dom(\tau_i) = \varnothing$ (can always bee guaranteed through a change of bound variables), and since $v \in Dom(\tau_i)$, by composition of two substitutions we have $v(\tau_i\varphi')_{|\mathcal{V}} = v\tau_i\varphi'$.
  - If v does not belong to $\mathcal{V}$ then by $\mathbf{H}_0$ we have $v \notin Dom(\tau_i)$. We want to show $v\varphi'(\tau_i\varphi')_{|\mathcal{V}} = v\tau_i\varphi'$. By the definition of substitution application, we have $v\tau_i\varphi' = v\varphi'$. Since $Dom(\tau_i\varphi')_{|\mathcal{V}} = \mathcal{V}$, we can rename bound variables so that $Dom(\tau_i\varphi')_{|\mathcal{V}} \cap \mathtt{fv}(Ran(\varphi')) = \varnothing$. By the definitions of substitution application and composition, we have $v\varphi'(\tau_i\varphi')_{|\mathcal{V}} = v\varphi'$.

□

Lemma 3 (Diamond Property of Parallel Reduction). *For all terms M, N, and Q, if $M \Rightarrow_{\beta_C} N$ and $M \Rightarrow_{\beta_C} Q$, then there exists a term W such that $N \Rightarrow_{\beta_C} W$ and $Q \Rightarrow_{\beta_C} W$.*

Proof. The proof is by induction on the structure of M.

- If $M = x$ or $M = f$ then the result holds since we necessarily have $M = N = Q$
- If M is an abstraction we consider the following cases:
  - if $M = \lambda_{\mathcal{V}} M_1.M_2$ then we have $N = \lambda_{\mathcal{V}} N_1.N_2$ with $M_1 \Rightarrow_{\beta_C} N_1, M_2 \Rightarrow_{\beta_C} N_2$ and $Q = \lambda_{\mathcal{V}} Q_1.Q_2$ with $M_1 \Rightarrow_{\beta_C} Q_1, M_2 \Rightarrow_{\beta_C} Q_2$. Applying the induction hypothesis to M_1 and to M_2 we get that there exist two terms W_1

and W_2 such that $N_1 \Rightarrow_{\beta_C} W_2, Q_1 \Rightarrow_{\beta_C} W_2$ and $N_2 \Rightarrow_{\beta_C} W_1, Q_2 \Rightarrow_{\beta_C} W_1$. So, by the definition of parallel reduction, we can conclude $N = \lambda_{\mathcal{V}} N_1.N_2 \Rightarrow_{\beta_C} \lambda_{\mathcal{V}} W_2.W_1 = W$ and $Q = \lambda_{\mathcal{V}} Q_1.Q_2 \Rightarrow_{\beta_C} \lambda_{\mathcal{V}} W_2.W_1 = W$.

- if $M = \lambda_{\mathcal{V}} M_1 + M_2.M_3$ then we have the following possible cases:
  1. $N = \lambda_{\mathcal{V}} N_1.N_3 + \lambda_{\mathcal{V}} N_2.N_3$ with $M_1 \Rightarrow_{\beta_C} N_1, M_2 \Rightarrow_{\beta_C} N_2, M_3 \Rightarrow_{\beta_C} N_3$ and $Q = \lambda_{\mathcal{V}} Q_1.Q_3 + \lambda_{\mathcal{V}} Q_2.Q_3$ with $M_1 \Rightarrow_{\beta_C} Q_1, M_2 \Rightarrow_{\beta_C} Q_2, M_3 \Rightarrow_{\beta_C} Q_3$. Applying the induction hypothesis to M_1, M_2 and M_3 we get that there exist terms W_1, W_2 and W_3 such that $N_1 \Rightarrow_{\beta_C} W_1, Q_1 \Rightarrow_{\beta_C} W_1, N_2 \Rightarrow_{\beta_C} W_2, Q_2 \Rightarrow_{\beta_C} W_2$ and $N_3 \Rightarrow_{\beta_C} W_3, Q_3 \Rightarrow_{\beta_C} W_3$. So, by the definition of parallel reduction, we can conclude $N = \lambda_{\mathcal{V}} N_1.N_3 + \lambda_{\mathcal{V}} N_2.N_3 \Rightarrow_{\beta_C} \lambda_{\mathcal{V}} W_1.W_3 + \lambda_{\mathcal{V}} W_2.W_3 = W$ and $Q = \lambda_{\mathcal{V}} Q_1.Q_3 + \lambda_{\mathcal{V}} Q_2.Q_3 \Rightarrow_{\beta_C} \lambda_{\mathcal{V}} W_1.W_3 + \lambda_{\mathcal{V}} W_2.W_3 = W$.
  2. $N = \lambda_{\mathcal{V}} N_1 + N_2.N_3$ with $M_1 \Rightarrow_{\beta_C} N_1, M_2 \Rightarrow_{\beta_C} N_2, M_3 \Rightarrow_{\beta_C} N_3$ and $Q = \lambda_{\mathcal{V}} Q_1.Q_3 + \lambda_{\mathcal{V}} Q_2.Q_3$ with $M_1 \Rightarrow_{\beta_C} Q_1, M_2 \Rightarrow_{\beta_C} Q_2, M_3 \Rightarrow_{\beta_C} Q_3$. Applying the induction hypothesis to M_1, M_2 and M_3 we get that there exist terms W_1, W_2 and W_3 such that $N_1 \Rightarrow_{\beta_C} W_1, Q_1 \Rightarrow_{\beta_C} W_1, N_2 \Rightarrow_{\beta_C} W_2, Q_2 \Rightarrow_{\beta_C} W_2$ and $N_3 \Rightarrow_{\beta_C} W_3, Q_3 \Rightarrow_{\beta_C} W_3$. So, by the definition of parallel we can conclude $N = \lambda_{\mathcal{V}} N_1 + N_2.N_3 \Rightarrow_{\beta_C} \lambda_{\mathcal{V}} W_1.W_3 + \lambda_{\mathcal{V}} W_2.W_3 = W$ and $Q = \lambda_{\mathcal{V}} Q_1.Q_3 + \lambda_{\mathcal{V}} Q_2.Q_3 \Rightarrow_{\beta_C} \lambda_{\mathcal{V}} W_1.W_3 + \lambda_{\mathcal{V}} W_2.W_3 = W$. Analogous for $N = \lambda_{\mathcal{V}} N_1.N_3 + \lambda_{\mathcal{V}} N_2.N_3$ and $Q = \lambda_{\mathcal{V}} Q_1 + Q_2.Q_3$
  3. The case for $N = \lambda_{\mathcal{V}} N_1 + N_2.N_3$ and $Q = \lambda_{\mathcal{V}} Q_1 + Q_2.Q_3$ is covered by the case for $M = \lambda_{\mathcal{V}} M_1.M_2$.
- if $M = \lambda_{\mathcal{V}} M_1.M_2 + M_3$ then we have the following cases:
  1. $N = \lambda_{\mathcal{V}} N_1.N_2 + \lambda_{\mathcal{V}} N_1.N_3$ with $M_1 \Rightarrow_{\beta_C} N_1, M_2 \Rightarrow_{\beta_C} N_2, M_3 \Rightarrow_{\beta_C} N_3$ and $Q = \lambda_{\mathcal{V}} Q_1.Q_2 + \lambda_{\mathcal{V}} Q_1.Q_3$ with $M_1 \Rightarrow_{\beta_C} Q_1, M_2 \Rightarrow_{\beta_C} Q_2, M_3 \Rightarrow_{\beta_C} Q_3$. Applying the induction hypothesis to M_1, M_2 and M_3 we get that there exist terms W_1, W_2 and W_3 such that $N_1 \Rightarrow_{\beta_C} W_1, Q_1 \Rightarrow_{\beta_C} W_1, N_2 \Rightarrow_{\beta_C} W_2, Q_2 \Rightarrow_{\beta_C} W_2$ and $N_3 \Rightarrow_{\beta_C} W_3, Q_3 \Rightarrow_{\beta_C} W_3$. So, by the definition of parallel reduction we can conclude $N = \lambda_{\mathcal{V}} N_1.N_2 + \lambda_{\mathcal{V}} N_1.N_3 \Rightarrow_{\beta_C} \lambda_{\mathcal{V}} W_1.W_2 + \lambda_{\mathcal{V}} W_1.W_3 = W$ and $Q = \lambda_{\mathcal{V}} Q_1.Q_2 + \lambda_{\mathcal{V}} Q_1.Q_3 \Rightarrow_{\beta_C} \lambda_{\mathcal{V}} W_1.W_2 + \lambda_{\mathcal{V}} W_1.W_3 = W$.
  2. if $M = \lambda_{\mathcal{V}} M_1.M_2 + M_3$ then we have $N = \lambda_{\mathcal{V}} N_1.N_2 + N_3$ with $M_1 \Rightarrow_{\beta_C} N_1, M_2 \Rightarrow_{\beta_C} N_2, M_3 \Rightarrow_{\beta_C} N_3$ and $Q = \lambda_{\mathcal{V}} Q_1.Q_2 + \lambda_{\mathcal{V}} Q_1.Q_3$ with $M_1 \Rightarrow_{\beta_C} Q_1, M_2 \Rightarrow_{\beta_C} Q_2, M_3 \Rightarrow_{\beta_C} Q_3$. Applying the induction hypothesis to M_1, M_2 and M_3 we get that there exist terms W_1, W_2 and W_3 such that $N_1 \Rightarrow_{\beta_C} W_1, Q_1 \Rightarrow_{\beta_C} W_1, N_2 \Rightarrow_{\beta_C} W_2, Q_2 \Rightarrow_{\beta_C} W_2$ and $N_3 \Rightarrow_{\beta_C} W_3, Q_3 \Rightarrow_{\beta_C} W_3$. So, by the definition of parallel reduction, we can conclude $N = \lambda_{\mathcal{V}} N_1.N_2 + N_3 \Rightarrow_{\beta_C} \lambda_{\mathcal{V}} W_1.W_2 + \lambda_{\mathcal{V}} W_1.W_3 = W$ and $Q = \lambda_{\mathcal{V}} Q_1.Q_2 + \lambda_{\mathcal{V}} Q_1.Q_3 \Rightarrow_{\beta_C} \lambda_{\mathcal{V}} W_1.W_2 + \lambda_{\mathcal{V}} W_1.W_3 = W$. Analogous for $N = \lambda_{\mathcal{V}} N_1.N_3 + \lambda_{\mathcal{V}} N_2.N_3$ and $Q = \lambda_{\mathcal{V}} Q_1.Q_2 + Q_3$
  3. The case for $N = \lambda_{\mathcal{V}} N_1.N_2 + N_3$ and $Q = \lambda_{\mathcal{V}} Q_1.Q_2 + Q_3$ is covered by the case for $M = \lambda_{\mathcal{V}} M_1.M_2$.

– If M is a term application then we have following cases:

- If $M = M_1X$ and $N = N_1X$ and $Q = Q_1X$ with $M_1 \Rightarrow_{\beta_C} N_1$ and $M_1 \Rightarrow_{\beta_C} Q_1$ then by induction hypothesis to M_1 there exist W_1 such that $N_1 \Rightarrow_{\beta_C} W_1, Q_1 \Rightarrow_{\beta_C} W_1$. By the definition of parallel reduction, we can conclude $N_1X \Rightarrow_{\beta_C} W_1X$ and $Q_1X \Rightarrow_{\beta_C} W_1X$.
- If $M = M_1 + M_2$ and $N = N_1 + N_2$ and $Q = Q_1 + Q_2$ with $M_1 \Rightarrow_{\beta_C} N_1, M_2 \Rightarrow_{\beta_C} N_2$ and $M_1 \Rightarrow_{\beta_C} Q_1, M_2 \Rightarrow_{\beta_C} Q_2$, then by induction hypothesis to M_1 and M_2 we get that there exist two terms W_1 and W_2 such that $N_1 \Rightarrow_{\beta_C} W_1, Q_1 \Rightarrow_{\beta_C} W_1$ and $N_2 \Rightarrow_{\beta_C} W_2, Q_2 \Rightarrow_{\beta_C} W_2$. We conclude that $N = N_1 + N_2 \Rightarrow_{\beta_C} W_1 + W_2$ and $Q = Q_1 + Q_2 \Rightarrow_{\beta_C} W_1 + W_2$.
- If $M = M_1M_2$ and $N = N_1N_2$ and $Q = Q_1Q_2$ with $M_1 \Rightarrow_{\beta_C} N_1, M_2 \Rightarrow_{\beta_C} N_2$ and $M_1 \Rightarrow_{\beta_C} Q_1, M_2 \Rightarrow_{\beta_C} Q_2$, then by induction hypothesis to M_1 and M_2 we get that there exist two terms W_1 and W_2 such that $N_1 \Rightarrow_{\beta_C} W_1, Q_1 \Rightarrow_{\beta_C} W_1$ and $N_2 \Rightarrow_{\beta_C} W_2, Q_2 \Rightarrow_{\beta_C} W_2$. We conclude that $N = N_1N_2 \Rightarrow_{\beta_C} W_1W_2$ and $Q = Q_1Q_2 \Rightarrow_{\beta_C} W_1W_2$.
- If $M = (\lambda_{\mathcal{V}} M_1.M_2)M_3$ we consider the following cases:
  1. $N = \sum_{i=1}^{n} N_2\varphi_i$ and $Q = \sum_{i=1}^{k} Q_2\varphi'_i$ with

$$\begin{cases} M_1 \Rightarrow_{\beta_C} N_1 \text{ and } M_1 \Rightarrow_{\beta_C} Q_1 \\ M_2 \Rightarrow_{\beta_C} N_2 \text{ and } M_2 \Rightarrow_{\beta_C} Q_2 \\ M_3 \Rightarrow_{\beta_C} N_3 \text{ and } M_3 \Rightarrow_{\beta_C} Q_3 \\ Sol(N_1 \lessdot_{\mathcal{V}} N_3) = \{\varphi_1, \ldots, \varphi_n\} \\ Sol(Q_1 \lessdot_{\mathcal{V}} Q_3) = \{\varphi'_1, \ldots, \varphi'_k\} \end{cases}$$

     then we apply the induction hypothesis on M_1, M_2 and M_3 and get

$$\begin{cases} N_1 \Rightarrow_{\beta_C} W_1 \text{ and } Q_1 \Rightarrow_{\beta_C} W_1 \\ N_2 \Rightarrow_{\beta_C} W_2 \text{ and } Q_2 \Rightarrow_{\beta_C} W_2 \\ N_3 \Rightarrow_{\beta_C} W_3 \text{ and } Q_3 \Rightarrow_{\beta_C} W_3 \end{cases}$$

     Applying $\mathbf{H}_2$ to N_1 and N_2, we get $Sol(W_1 \lessdot_{\mathcal{V}} W_3) = \{\varphi''_1, \ldots, \varphi''_m\}$ where each φ_i reduces with the parallel reduction to at least one φ''_j and every φ''_j is the parallel reduction of some φ_i. Then by applying Lemma 2 we conclude that $N_2\varphi_i \Rightarrow_{\beta_C} W_2\varphi''_j$, by the definition of parallel reduction and by the ACID property of $+$, we have $\sum_{i=1}^{n} N_2\varphi_i \Rightarrow_{\beta_C} \sum_{i=1}^{n} W_2\varphi''_i$. Again by applying $\mathbf{H}_2$ to Q_1 and Q_2 we have $Sol(W_1 \lessdot_{\mathcal{V}} W_3) = \{\varphi''_1, \ldots, \varphi''_m\}$ with each φ_i reduces with parallel reduction to at least one φ''_j and every φ''_j is the parallel reduction of some φ_i. By Lemma 2 we can conclude that $Q_2\varphi'_i \Rightarrow_{\beta_C} W_2\varphi''_j$, therefore by the definition of parallel reduction and by the ACID property of $+$, we have $\sum_{i=1}^{k} Q_2\varphi'_i \Rightarrow_{\beta_C} \sum_{i=1}^{n} W_2\varphi''_i$.
  2. $N = \sum_{i=1}^{n} N_2\varphi_i$ and $Q = (\lambda_{\mathcal{V}} Q_1.Q_2)Q_3$ with

$$\begin{cases} M_1 \Rightarrow_{\beta_C} N_1 \text{ and } M_1 \Rightarrow_{\beta_C} Q_1 \\ M_2 \Rightarrow_{\beta_C} N_2 \text{ and } M_2 \Rightarrow_{\beta_C} Q_2 \\ M_3 \Rightarrow_{\beta_C} N_3 \text{ and } M_3 \Rightarrow_{\beta_C} Q_3 \\ Sol(N_1 \lessdot_{\mathcal{V}} N_3) = \{\varphi_1, \ldots, \varphi_n\} \end{cases}$$

then we apply the induction hypothesis on M_1, M_2 and M_3 and get

$$\begin{cases} N_1 \Rightarrow_{\beta_C} W_1 \text{ and } Q_1 \Rightarrow_{\beta_C} W_1 \\ N_2 \Rightarrow_{\beta_C} W_2 \text{ and } Q_2 \Rightarrow_{\beta_C} W_2 \\ N_3 \Rightarrow_{\beta_C} W_3 \text{ and } Q_3 \Rightarrow_{\beta_C} W_3 \end{cases}$$

By applying $\mathbf{H}_2$ we have $Sol(W_1 \lessdot_{\mathcal{V}} W_3) = \{\varphi''_1, \ldots, \varphi''_m\}$ with each φ_i reduces with parallel reduction to at least one φ''_j and every φ''_j is the parallel reduction of some φ_i. Then by applying Lemma 2 we conclude that $N_2\varphi_i \Rightarrow_{\beta_C} W_2\varphi''_j$ and by the definition of parallel reduction and by the ACID property of $+$, we have $\sum_{i=1}^{n} N_2\varphi_i \Rightarrow_{\beta_C} \sum_{i=1}^{m} W_2\varphi''_i$. In other hand we have $(\lambda_{\mathcal{V}} Q_1.Q_2)Q_3 \Rightarrow_{\beta_C} \sum_{i=1}^{m} W_2\varphi''_i$. Analogous for $N = (\lambda_{\mathcal{V}} N_1.N_2)N_3$ and $Q = \sum_{i=1}^{n} Q_2\varphi_i$

3. The case for $N = (\lambda_{\mathcal{V}} N_1.N_2)N_3$ and $Q = (\lambda_{\mathcal{V}} Q_1.Q_2)Q_3$ is covered by the case for $M = M_1M_2$.

□

Theorem 1. *For all terms M, N, and Q, if $M \twoheadrightarrow_{\beta_C} N$ and $M \twoheadrightarrow_{\beta_C} Q$ then there exists a term W such that $N \twoheadrightarrow_{\beta_C} W$ and $Q \twoheadrightarrow_{\beta_C} W$.*

Proof. From Lemma 1 we get that the reflexive-transitive closure of the relations $\rightarrow_{\beta_C}$ and $\Rightarrow_{\beta_C}$ are the same. By Lemma 3, the relation $\Rightarrow_{\beta_C}$ has the diamond property and, therefore, its reflexive-transitive closure is confluent. Hence, we conclude that the relation $\twoheadrightarrow_{\beta_C}$ is confluent.

This theorem shows that under the condition of *Sol* satisfying $\mathbf{H}_0$, $\mathbf{H}_1$, and $\mathbf{H}_2$, the relation $\twoheadrightarrow_{\beta_C}$ (and, hence, the calculus) is confluent.

Example 5. Looking back to Example 4, is it not surprising that confluence does not hold there: *Sol* used in that example violates the $\mathbf{H}_1$ property. Just take $P = f(X, Y), M = f(Z), \mathcal{V} = \{X, Y\}$, and $\vartheta = \{Z \mapsto \langle a, b\rangle\}$. Then we get $Sol(P \lessdot_{\mathcal{V}} M) = \{\{X \mapsto \epsilon, Y \mapsto Z\}, \{X \mapsto Z, Y \mapsto \epsilon\}\}$ and $Sol(P\vartheta \lessdot_{\mathcal{V}} M\vartheta) = \{\{X \mapsto \epsilon, Y \mapsto \langle a, b\rangle\}, \{X \mapsto a, Y \mapsto b\}, \{X \mapsto \langle a, b\rangle, Y \mapsto \epsilon\}\}$.

We now define *Sol* slightly differently from that in Example 3. In particular, it should satisfy all the conditions from Example 3 and, in addition, one more:

C3. If M contains a hedge variable, then $Sol(P \lessdot_{\mathcal{V}} M)$ is undefined.

This *Sol* satisfies $\mathbf{H}_0$, $\mathbf{H}_1$, and $\mathbf{H}_2$ (and, thus, it defines a non-trivial matching algorithm with hedge variables which makes the calculus confluent). The conditions trivially hold when $Sol(P \lessdot_{\mathcal{V}} M)$ is undefined. Assume $Sol(P \lessdot_{\mathcal{V}} M)$ is defined. Then, by condition C2, it returns a nonempty set of substitutions. $\mathbf{H}_0$ follows from the facts that $\mathtt{fv}(P) = \mathcal{V}$ (condition C1) and $\mathtt{fv}(M) \cap \mathcal{V} = \varnothing$. $\mathbf{H}_1$ is implied by C3, because the absence of hedge variables in M guarantees that the structure of P and Q are maintained in $P\vartheta$ and $Q\vartheta$: For each subterm M' of M, $M'\vartheta$ is again a single term. If an x from P matched M' with $x\varphi = M'$, now we have $x\varphi\vartheta = M'\vartheta$. The same holds for an X from P: In $M\vartheta$, $X\varphi\vartheta$ will match the instance of the same hedge, to which $X\varphi$ matched in M. As for $\mathbf{H}_2$, it trivially holds because P and M do not contain λ-abstractions and sums. This solving algorithm makes the first reduction in Example 4 impossible.

3 The Typed Calculus

In this section, we introduce types in our calculus and show that the Subject Reduction and Strong Normalization properties hold.

3.1 The Type System

Let $\mathbb{A}$ be a set of type atoms. The set of *term types* over $\mathbb{A}$, denoted $\mathbb{T}_{\mathbb{A}}$ or simply $\mathbb{T}$, is defined inductively: $\alpha \in \mathbb{A} \Rightarrow \alpha \in \mathbb{T}$, $\tau_1, \tau_2 \in \mathbb{T} \Rightarrow (\tau_1 \to \tau_2) \in \mathbb{T}$, and $\tau_1, \tau_2 \in \mathbb{T} \Rightarrow (\tau_1^* \to \tau_2) \in \mathbb{T}$. The set of *star types* (over $\mathbb{A}$), denoted $\mathbb{T}^\star_{\mathbb{A}}$ or simply $\mathbb{T}^*$, is defined as $\tau \in \mathbb{T} \Rightarrow \tau \in \mathbb{T}^*$ and $\tau \in \mathbb{T} \Rightarrow \tau^* \in \mathbb{T}^*$.

Note that if a type does not contain a star type, then it is a standard simple type. We denote the set of simple types by $\mathbb{S}$. The letter τ will be used to denote elements of $\mathbb{T}$, the letter ρ for elements of $\mathbb{T}^*$, σ for elements of $\mathbb{S}$. As usual, $\rho_1 \to \cdots \to \rho_n \to \tau$ stands for $(\rho_1 \to (\rho_2 \to \cdots \to (\rho_n \to \tau) \cdots))$.

The *subtyping relation* is the preorder generated by the relation $\leqslant$ defined as:

$$\tau^* \to \tau' \leqslant \tau' \qquad \tau^* \to \tau' \leqslant \tau^* \to \tau^* \to \tau' \qquad \tau \leqslant \tau^*$$
$$\tau_1^* \leqslant \tau_2^* \text{ if } \tau_1 \leqslant \tau_2 \qquad \rho_1 \to \tau_1 \leqslant \rho_2 \to \tau_2 \text{ if } \rho_2 \leqslant \rho_1 \text{ and } \tau_1 \leqslant \tau_2$$

We denote the subtyping relation with $\leqslant$ as well. Notice that simple types are the maximal types with respect to this relation. Also, two distinct simple types are not comparable with respect to $\leqslant$.

We assume that each $f \in \mathcal{F}$ has the unique associated type, $\mathit{type}(f) \in \mathbb{T}$. A *type assignment statement* is an expression of the form $x : \sigma$, $X : \sigma^*$, and $M : \tau$ with $x \in \mathcal{X}_{\mathrm{t}}$, $X \in \mathcal{X}_{\mathrm{h}}$, $M \in \mathcal{T} \backslash \mathcal{X}_{\mathrm{t}}$, $\sigma \in \mathbb{S}$, $\tau \in \mathbb{T}$. The types σ, τ, τ^* are the *predicate* and x, X, M are the *subject* of the statement.

A *declaration* is a statement whose subject is a term variable or a hedge variable. A *basis* Γ is a set of declarations $\Gamma = \{x_1 : \sigma_1, \ldots, x_n : \sigma_n, X_1 : \sigma_1'^*, \ldots, X_m : \sigma_m'^*\}$, where all x's and X's are distinct.

By $\mathit{Subject}(\Gamma)$ we denote the set of variables that are subjects of the statements in Γ: $\mathit{Subject}(\Gamma) = \{v \mid v : \rho \in \Gamma\}$.

Note that the variables are assigned simple types or starred simple types, while constants may have an arbitrary type from $\mathbb{T}$.

A statement $s : \rho$ is *derivable* from a basis Γ, written $\Gamma \vdash s : \rho$, if $\Gamma \vdash s : \rho$ can be produced by the following rules:

$$\frac{}{\Gamma, x : \sigma \vdash x : \sigma}\,(\mathrm{IV}) \qquad \frac{}{\Gamma, X : \sigma^* \vdash X : \sigma^*}\,(\mathrm{HV})$$

$$\frac{}{\Gamma \vdash f : \mathit{type}(f)}\,(\mathrm{FUN}) \qquad \frac{\Gamma \vdash M : \rho \to \tau \quad \Gamma \vdash s : \rho}{\Gamma \vdash Ms : \tau}\,(\mathrm{APP})$$

$$\frac{\Gamma \vdash M : \tau \quad \Gamma \vdash N : \tau}{\Gamma \vdash M + N : \tau}\,(\mathrm{SUM}) \qquad \frac{\Gamma \vdash M : \tau_1 \quad \tau_1 \leqslant \tau_2}{\Gamma \vdash M : \tau_2}\,(\mathrm{SUB})$$

$$\frac{\Gamma, \Delta \vdash M : \tau_1 \qquad \Gamma, \Delta \vdash N : \tau_2 \qquad \mathit{Subject}(\Delta) = \mathcal{V}}{\Gamma \vdash \lambda_{\mathcal{V}} M.N : \tau_1 \to \tau_2}\,(\mathrm{ABS})$$

The typing relation extends to hedges by the following typing rule:

$$\frac{\Gamma \vdash s_1 : \rho_1, \ldots, \Gamma \vdash s_n : \rho_n \qquad \rho_1 \leqslant \rho, \ldots, \rho_n \leqslant \rho}{\Gamma \vdash \langle s_1, \ldots, s_n \rangle : \rho} (\text{HED})$$

The notion of substitution extends to the typed setting in the usual way, requiring that the substitutions preserve types. In particular, we say that substitution φ is *well-typed* in a basis Γ if and only if for all $v \in \mathit{Dom}(\varphi)$, there exist types ρ_1 and ρ_2 with $\rho_2 \leqslant \rho_1$ such that $v : \rho_1 \in \Gamma$ and $\Gamma \vdash v\varphi : \rho_2$. In this case, we write φ_Γ to indicate that φ is well-typed in the basis Γ.

From now on assume that the substitutions computed by *Sol* are well-typed. An example of such a solving algorithm would be a reformulation of the hedge matching algorithm from Example 5 in the typed setting. The simplest such reformulation would be to add an additional condition, which requires from *Sol* to keep only well-typed substitutions from the computed ones. (Such an approach is quite common in unification, producing solutions in the unsorted setting and keeping only the well-sorted/typed ones, see, e.g., [32, 28, 33].)

Two properties we would like to prove for the typed version of our calculus are Subject Reduction (SR) and Strong Normalization (SN). Subject Reduction means that types are preserved under reduction. Strong Normalization means that there are no infinite reduction chains. Comparing to the proofs from the lambda calculus, we have to show that the handling of star types and the nondeterminism represented by sums are done properly. Note that SN does not hold in the untyped calculus. Even more, it would not hold if we had not restricted the types of variables to be constructed over simple types (the same holds for the SR property). We will see the corresponding counterexamples below.

3.2 Subject Reduction

Subject reduction is based on two lemmas: The Generation Lemma and the Substitution Lemma.

Lemma 4 (Generation Lemma).

- *If* $\Gamma \vdash x : \sigma$*, then* $(x : \sigma') \in \Gamma$*, and* $\sigma' \leqslant \sigma$*.*
- *If* $\Gamma \vdash X : \sigma*$*, then* $(X : \sigma_1^*) \in \Gamma$*, and* $\sigma_1^* \leqslant \sigma^*$*.*
- *If* $\Gamma \vdash Ms : \sigma$*, then there exists* ρ *such that* $\Gamma \vdash M : \rho \to \tau$ *and* $\Gamma \vdash s : \rho$*, and* $\tau \leqslant \sigma$*.*
- *If* $\Gamma \vdash \lambda_{\mathcal{V}} M.N : \sigma$*, then there exist* τ_1, τ_2, Δ *such that* $\tau = \tau_1 \to \tau_2$*,* $\mathit{Subject}(\Delta) = \mathcal{V}$*,* $\Gamma, \Delta \vdash M : \tau_1$ *and* $\Gamma, \Delta \vdash N : \tau_2$*, and* $\tau \leqslant \sigma$*.*
- *If* $\Gamma \vdash M + N : \sigma$*, then* $\Gamma \vdash M : \tau$ *and* $\Gamma \vdash N : \tau$*, and* $\tau \leqslant \sigma$*.*

Proof. By induction of the length of derivation.

- If the derivation of $\Gamma \vdash x : \sigma$ is by rule IV then we have $(x : \sigma) \in \Gamma$ and $\sigma \leqslant \sigma$. If $\Gamma \vdash x : \sigma$ is by rule SUB then $\Gamma \vdash x : \sigma'$ and $\sigma' \leqslant \sigma$. By induction hypothesis $x : \sigma'' \in \Gamma$ and $\sigma'' \leqslant \sigma' \leqslant \sigma$.

- If the derivation of $\Gamma \vdash X : \sigma^*$ is by rule HV then we have $(X : \sigma^*) \in \Gamma$, and $\sigma^* \leqslant \sigma^*$. If the derivation of $\Gamma \vdash X : \tau^*$ is by rule SUB then we have $\Gamma \vdash X : \tau'^*$ and $\tau'^* \leqslant \tau^*$, by induction hypothesis we get $(X : \tau'^*) \in \Gamma$, and $\tau'^* \leqslant \tau^* \leqslant \tau^*$.
- If the derivation of $\Gamma \vdash Ms : \sigma$ is by rule APP then there exist ρ' such that $\Gamma \vdash M : \rho' \to \sigma$ and $\Gamma \vdash s : \rho'$. So, we can conclude $\Gamma \vdash M : \rho' \to \tau$ and $\Gamma \vdash s : \rho'$, with $\tau = \sigma$. If the derivation of $\Gamma \vdash Ms : \sigma$ is by rule SUB then we have $\Gamma \vdash Ms : \sigma'$ and $\sigma' \leqslant \sigma$. By induction hypothesis there is ρ, τ such that $\Gamma \vdash M : \rho \to \tau$, $\Gamma \vdash s : \rho$ and $\tau \leqslant \sigma'$. Therefore $\Gamma \vdash M : \rho \to \tau$ and $\Gamma \vdash s : \rho$ and $\tau \leqslant \sigma' \leqslant \sigma$.
- If the derivation of $\Gamma \vdash \lambda_{\mathcal{V}} M.N : \sigma$ is by rule ABS then there exist τ_1, τ_2, Δ such that $\sigma = \tau_1 \to \tau_2$, $Subject(\Delta) = \mathcal{V}$, $\Gamma, \Delta \vdash M : \tau_1$ and $\Gamma, \Delta \vdash N : \tau_2$. The result follows considering $\tau = \sigma$. If the derivation of $\Gamma \vdash \lambda_{\mathcal{V}} M.N : \sigma$ is by rule SUB then we have $\Gamma \vdash \lambda_{\mathcal{V}} M.N : \sigma'$ and $\sigma' \leqslant \sigma$. By induction hypothesis there is $\tau, \tau_1, \tau_2, \Delta$ such that $\tau = \tau_1 \to \tau_2$, $\Gamma, \Delta \vdash M : \tau_1$ and $\Gamma, \Delta \vdash N : \tau_2$ and $\tau \leqslant \sigma'$. The result follows with $\tau \leqslant \sigma' \leqslant \sigma$.
- If the derivation of $\Gamma \vdash M + N : \sigma$ is by rule SUM then $\Gamma \vdash M : \sigma$ and $\Gamma \vdash N : \sigma$. The result follows considering $\tau = \sigma$. If the derivation of $\Gamma \vdash M + N : \tau$ is by rule SUB then we have $\Gamma \vdash M + N : \sigma'$ and $\sigma' \leqslant \sigma$. By induction hypothesis $\Gamma \vdash M : \tau$ and $\Gamma \vdash N : \tau$ and $\tau \leqslant \sigma'$. Therefore we get $\Gamma \vdash M : \tau$ and $\Gamma \vdash N : \tau$ and $\tau \leqslant \sigma' \leqslant \sigma$.

□

Lemma 5 (Substitution Lemma). *Let $\varphi = \{v_1 \mapsto h_1, \ldots v_n \mapsto h_n\}$ be a well-typed substitution with respect to $\Gamma, v_1 : \tau_1, \ldots, v_n : \tau_n$. If $\Gamma, v_1 : \tau_1, \ldots, v_n : \tau_n \vdash M : \sigma$ and for all $1 \leqslant i \leqslant n$ $\Gamma \vdash h_i : \tau_i'$, with $\tau_i' \leqslant \tau_i$, then $\Gamma \vdash M\varphi : \tau$ and $\tau \leqslant \sigma$.*

Proof. By structural induction on M:

- If $M = x$ then we consider two cases: if $x \notin Dom(\varphi)$ then $M\varphi = x$ and $\Gamma \vdash x : \sigma$; if $x \in Dom(\varphi)$, then by Lemma 4, $x : \sigma' \in \Gamma$ and $\sigma' \leqslant \sigma$. Let $x = x_i$, then by Lemma 4, $x_i : \tau_i \in \Gamma$ and $\tau_i \leqslant \sigma$ and $x\varphi = N_i$, therefore $\Gamma \vdash N_i : \tau_i'$ and $\tau_i' \leqslant \tau_i \leqslant \sigma$.
- If M is f then $f = f\varphi$, therefore the result follows trivially.
- If M is $M_1 M_2$ then by Lemma 4 there exist ρ, τ such that $\Gamma, x_1 : \tau_1, \ldots, x_n : \tau_n \vdash M_1 : \rho \to \tau$ and $\Gamma, x_1 : \tau_1, \ldots, x_n : \tau_n \vdash M_2 : \rho$ and $\tau \leqslant \sigma$. By induction hypothesis $\Gamma \vdash M_1\varphi : \sigma'$ and $\sigma' \leqslant \rho' \to \tau'$ (which implies $\sigma' = \rho'' \to \tau''$ and $\rho \leqslant \rho'$, $\tau' \leqslant \tau$) and $\Gamma \vdash M_2\varphi : \rho''$, with $\rho'' \leqslant \rho$. Therefore $\Gamma \vdash M_2\varphi : \rho'$, since $\rho'' \leqslant \rho \leqslant \rho'$, which implies $\Gamma \vdash (M_1 M_2)\varphi : \tau'$, therefore, since $\tau' \leqslant \tau$, $\Gamma \vdash (M_1 M_2)\varphi : \tau$ and $\tau \leqslant \sigma$.
- If M is $M_1 X$ then by Lemma 4 there exist ρ^*, τ such that $\Gamma, x_1 : \tau_1, \ldots, x_n : \tau_n \vdash M_1 : \rho^* \to \tau$ and $\Gamma, x_1 : \tau_1, \ldots, x_n : \tau_n \vdash X : \rho^*$, $\rho''^* \in \Gamma, x_1 : \tau_1, \ldots, x_n : \tau_n$, with $\rho''^* \leqslant \rho^*$ and $\tau \leqslant \sigma$. We consider two cases:
  1. $\varphi(X) = \epsilon$, in which case $(M_1 X)\varphi = M_1\varphi$. By induction hypothesis we have $\Gamma \vdash M_1\varphi : \rho'^* \to \tau'$ and $\rho^* \leqslant \rho'^*$, $\tau' \leqslant \tau$. Since $\rho'^* \to \tau' \leqslant \tau' \leqslant \tau$, one concludes $\Gamma \vdash M_1\varphi : \tau$ with $\tau \leqslant \sigma$.

2. $\varphi(X) = \epsilon$, in which case $(M_1X)\varphi = ((M_1\varphi)s_1 \dots s_n)$. By induction hypothesis we have $\Gamma \vdash M_1\varphi : \rho'^* \to \tau'$ and $\rho^* \leqslant \rho'^*,\ \tau' \leqslant \tau$. Since $(s_1 : \rho_1, \dots, s_n : \rho_n) : \rho''^*$ one has $\rho_i \leqslant \rho''^*, 1 \leqslant i \leqslant n$ and $\rho''^* \leqslant \rho^*$. By SUB rule one can derive $\Gamma \vdash M_1\varphi : \underbrace{\rho'^* \to \dots \to \rho'^*}_{n} \to \tau'$. Since $\rho_1 \leqslant \rho''^* \leqslant \rho^* \leqslant \rho'^*$, by the contravariance rulewe have $\underbrace{\rho'^* \to \dots \to \rho'^*}_{n} \to \tau' \leqslant \rho_1 \to \underbrace{\rho'^* \to \dots \to \rho'^*}_{n-1} \to \tau'$, so by SUB we have $\Gamma \vdash M_1\varphi : \rho_1 \to \underbrace{\rho'^* \to \dots \to \rho'^*}_{n-1} \to \tau'$ and by APP rule we finally have $\Gamma \vdash M_1\varphi s_1 : \underbrace{\rho'^* \to \dots \to \rho'^*}_{n-1} \to \tau'$. Finite application of the process described above leads to $\Gamma \vdash M_1\varphi s_1 \dots s_n = (M_1X)\varphi : \tau'$ and since $\tau' \leqslant \tau$ one concludes $\Gamma \vdash (M_1X)\varphi : \tau$, with $\tau \leqslant \sigma$.

– M is $\lambda_{\mathcal{V}} M_1.M_2$ then by Lemma 4 there exist τ_1, τ_2, Δ such that $\tau = \tau_1 \to \tau_2$, $Subject(\Delta) = \mathcal{V}$ and $\Gamma, x_1 : \tau_1, \dots, x_n : \tau_n, \Delta \vdash M_1 : \tau_1, \Gamma, x_1 : \tau_1, \dots, x_n : \tau_n, \Delta \vdash M_2 : \tau_2$ and $\tau_1 \to \tau_2 \leqslant \sigma$. By induction hypothesis $\Gamma, \Delta \vdash M_1\varphi : \tau_1'$ and $\Gamma, \Delta \vdash M_2\varphi : \tau_2'$ with $\tau_1' \leqslant \tau_1,\ \tau_2' \leqslant \tau_2$ and $Subject(\Delta) = \mathcal{V}$. Since $\tau_1' \leqslant \tau_1$ implies $\Gamma, \Delta \vdash M_1\varphi : \tau_1$ and $\tau_2' \leqslant \tau_2$ implies $\Gamma, \Delta \vdash M_2\varphi : \tau_2$. So, we get $\Gamma \vdash \lambda_{\mathcal{V}} M_1\varphi.M_2\varphi : \tau_1 \to \tau_2$ and $\tau_1 \to \tau_2 \leqslant \sigma$.

– If M is $M_1 + M_2$ then by Lemma 4 we have $\Gamma, x_1 : \tau_1, \dots, x_n : \tau_n \vdash M_1 : \tau$ and $\Gamma, x_1 : \tau_1, \dots, x_n : \tau_n \vdash M_2 : \tau$ and $\tau \leqslant \sigma$. By induction hypothesis $\Gamma \vdash M_1\varphi : \tau'$, $\Gamma \vdash M_2\varphi : \tau''$ and $\tau' \leqslant \tau,\ \tau'' \leqslant \tau$. Therefore $\Gamma \vdash M_1\varphi : \tau$ and $\Gamma \vdash M_2\varphi : \tau$, thus we can conclude $\Gamma \vdash M_1\varphi + M_2\varphi : \tau$ with $\tau \leqslant \sigma$.

□

These two lemmas imply the SR property of the typed version of the calculus.

Theorem 2 (Subject Reduction). *theorem If $s \twoheadrightarrow_{\beta_C} s'$, then $\Gamma \vdash s : \rho \Rightarrow \Gamma \vdash s' : \rho$.*

Proof. Suppose $\Gamma \vdash s : \rho$ and $s \to_{\beta_C} s'$ in order to show that $\Gamma \vdash s' : \rho$, then the result follows by induction on the derivation of $\Gamma \vdash s : \rho$.

Case 1. If s is either x, X or f then it is in normal form with respect to $\to_{\beta_C}$, so the result follows trivially.

Case 2. Assume $\Gamma \vdash s = NQ : \rho$. By Lemma 4 there exist τ', τ such that $\tau \leqslant \rho$ and $\Gamma \vdash N : \tau' \to \tau$ and $\Gamma \vdash Q : \tau'$. We have the following subcases:

Subcase 2.1. $s' = N'Q$ with $N \to_{\beta_C} N'$: By the induction hypothesis and by the SUB rule we have $\Gamma \vdash s' : \rho$.

Subcase 2.2. $s' = NQ'$ with $Q \to_{\beta_C} Q'$. Similar to the Subcase 2.1.

Subcase 2.3. If $\Gamma \vdash (\lambda_{\mathcal{V}} M_1.M_2)Q : \rho$ by Lemma 4 there exist τ_1 such that $\Gamma \vdash (\lambda_{\mathcal{V}} M_1.M_2) : \tau_1 \to \tau$ and $\Gamma \vdash Q : \tau_1$ and $\tau \leqslant \rho$. By Lemma 4 there exist τ_1', τ', Δ such that $Subject(\Delta) = \mathcal{V}, \Gamma, \Delta \vdash M_1 : \tau_1'$ and $\Gamma, \Delta \vdash M_2 : \tau'$ and $\tau' \leqslant \tau,\ \tau_1 \leqslant \tau_1'$. If $Sol(M_1 \lessdot_{\mathcal{V}} Q) = \{\varphi_1, \dots, \varphi_n\}$ then $(\lambda_{\mathcal{V}} M_1.M_2)Q \to_{\beta_C} \sum_{i=1}^{n} M_2\varphi_i$. By Lemma 5 (considering that

$Subject(\Delta) = \mathcal{V}$) we have $\Gamma \vdash M_2\varphi_i : \tau_i$ with $\tau_i \leqslant \tau' \leqslant \tau \leqslant \rho$, therefore $\Gamma \vdash \sum_{i=1}^{n} M_2\varphi_i : \rho$.

Case 3. Assume $\Gamma \vdash s = \lambda_{\mathcal{V}} N.Q : \rho$. By Lemma 4 there exist τ_1, τ_2, Δ such that $\tau_1 \to \tau_2 \leqslant \rho, Subject(\Delta) = \mathcal{V}$ and $\Gamma, \Delta \vdash N : \tau_1$ and $\Gamma, \Delta \vdash Q : \tau_2$.

Subcase 3.1. $s' = \lambda_{\mathcal{V}} N'.Q$ with $N \to_{\beta_C} N'$. By the induction hypothesis we have $\Gamma \vdash s' : \tau_1 \to \tau_2$, therefore since $\tau_1 \to \tau_2 \leqslant \rho$, $\Gamma \vdash s' : \rho$.

Subcase 3.2. $s' = \lambda_{\mathcal{V}} N.Q'$ with $Q \to_{\beta_C} Q'$. Similar to the Subcase 3.1.

Subcase 3.3. If $\Gamma \vdash \lambda_{\mathcal{V}} M_1 + M_2.M_3 : \rho$ by Lemma 4 there exist τ_1, τ_2, Δ such that $\tau_1 \to \tau_2 \leqslant \rho, Subject(\Delta) = \mathcal{V}$ and $\Gamma, \Delta \vdash M_1 + M_2 : \tau_1$ and $\Gamma, \Delta \vdash M_3 : \tau_2$. By Lemma 4 again we get $\Gamma, \Delta \vdash M_1 : \tau_1'$ and $\Gamma, \Delta \vdash M_2 : \tau_1'$ and $\tau_1' \leqslant \tau_1$. By SUB and $\tau_1' \leqslant \tau_1$ we have $\Gamma, \Delta \vdash M_1 : \tau_1$ and $\Gamma, \Delta \vdash M_2 : \tau_1$, and by ABS rule we have $\Gamma \vdash \lambda_{\mathcal{V}} M_1.M_3 : \tau_1 \to \tau_2$ and $\Gamma \vdash \lambda_{\mathcal{V}} M_2.M_3 : \tau_1 \to \tau_2$. By SUM rule we get $\Gamma \vdash \lambda_{\mathcal{V}} M_1.M_3 + \lambda_{\mathcal{V}} M_2.M_3 : \tau_1 \to \tau_2$, therefore, since $\tau_1 \to \tau_2 \leqslant \rho$, $\Gamma \vdash \lambda_{\mathcal{V}} M_1.M_3 + \lambda_{\mathcal{V}} M_2.M_3 : \rho$.

Subcase 3.4. If $\Gamma \vdash \lambda_{\mathcal{V}} M_1.M_2 + M_3 : \rho$ by Lemma 4 there exist τ_1, τ_2, Δ such that $\tau_1 \to \tau_2 \leqslant \rho, Subject(\Delta) = \mathcal{V}$ and $\Gamma, \Delta \vdash M_1 : \tau_1$ and $\Gamma, \Delta \vdash M_2 + M_3 : \tau_2$ and by Lemma 4 again we get $\Gamma, \Delta \vdash M_2 : \tau_2'$ and $\Gamma, \Delta \vdash M_3 : \tau_2'$ and $\tau_2' \leqslant \tau_2$. By typing rules first we have $\Gamma \vdash \lambda_{\mathcal{V}} M_1.M_2 : \tau_1 \to \tau_2'$ and $\Gamma \vdash \lambda_{\mathcal{V}} M_1.M_3 : \tau_1 \to \tau_2'$ and since $\tau_2' \leqslant \tau_2$, we have $\tau_1 \to \tau_2' \leqslant \tau_1 \to \tau_2$, therefore, $\Gamma \vdash \lambda_{\mathcal{V}} M_1.M_2 + \lambda_{\mathcal{V}} M_1.M_3 : \tau_1 \to \tau_2$ which implies $\Gamma \vdash \lambda_{\mathcal{V}} M_1.M_2 + \lambda_{\mathcal{V}} M_1.M_3 : \rho$.

□

Example 6. If variables were permitted to have arbitrary types instead of simple types or starred simple types, then the Subject Reduction theorem would not hold: Assume $(x : \tau^* \to \tau) \in \Gamma$ and $type(a) = \tau$. Then we have $\Gamma \vdash x : \tau \to \tau$, $\Gamma \vdash x : \tau$, $\Gamma \vdash \lambda x.(x\,x) : \tau \to \tau$ and, finally, $\Gamma \vdash (\lambda x.(x\,x)a) : \tau$. However, $(a\,a)$, that is obtained by reducing $(\lambda x.(x\,x)\,a)$, is not typeable anymore.

3.3 Strong Normalization

We concentrate now on the property of Strong Normalization of well-typed terms, using techniques from [31]. This property guarantees the existence of a normal form and in our case, as the calculus is confluent, we can also conclude the uniqueness of normal forms.

We start with defining some technical notions. First, a term is called *strongly normalizing* if there is no infinite reduction starting from it. We denote the *set of strongly normalizing terms* by SN. A *saturated set* S is a subset of SN, satisfying the following conditions:

1. For all $m \geqslant 0$ and for all $Q_1, \ldots, Q_m \in SN$, we have $xQ_1 \cdots Q_m \in S$ for all x of the corresponding type.
2. For all $m \geqslant 0$, for all $Q_1, \ldots, Q_m \in SN$, and for all $P, M \in SN$, if $Sol(P \lessdot_{\mathcal{V}} M) = \{\varphi_1, \ldots, \varphi_n\}$ and $N\varphi_1 Q_1 \cdots Q_m + \cdots + N\varphi_n Q_1 \cdots Q_m \in S$, then $(\lambda_{\mathcal{V}} P.N)MQ_1 \cdots Q_m \in S$.

The *set of all saturated sets* is denoted by SAT.

Next, we interpret types in the set of terms. Given a type ρ, its *interpretation* $[\![\rho]\!]$ is a set of terms defined as follows:

$$\begin{aligned}
[\![\alpha]\!] &= SN.\\
[\![\tau{\to}\tau']\!] &= \{W \mid WQ \in [\![\tau']\!] \text{ for all } Q \in [\![\tau]\!]\}.\\
[\![\tau^*{\to}\tau']\!] &= \{W \mid WQ_1 \cdots Q_m \in [\![\tau']\!] \text{for all } m \geqslant 0 \text{ and } Q_1, \ldots, Q_m \in [\![\tau]\!]\}.\\
[\![\tau^*]\!] &= \{\langle Q_1, \ldots, Q_m\rangle \mid \text{for all } m \geqslant 0 \text{ and for all } Q_1, \ldots, Q_m \in [\![\tau]\!]\}.
\end{aligned}$$

The SN property is based on several lemmas:

Lemma 6 (Types are Saturated Sets).

1. *$SN \in SAT$.*
2. *If $S_1, S_2 \in SAT$, then $\{W \mid WQ \in S_2$ for all $Q \in S_1\} \in SAT$.*
3. *If $S_1, S_2 \in SAT$ then $\{W \mid WQ_1 \ldots Q_n \in S_2$ for all $n \geqslant 0$ and for all terms $Q_1, \ldots, Q_n \in S_1\} \in SAT$.*
4. *If $\{S_i\}_{i\in I}$ is a collection of members of SAT, then $\cap_{i\in I} S_i \in SAT$.*
5. *$[\![\tau]\!] \in SAT$ for all types τ.*

Proof. 1. One has $SN \subseteq SN$ and satisfies condition (a) in the definition of saturation. As to condition (b), suppose $\sum_{i=1}^{n} N\varphi_i Q_1 \ldots Q_m \in SN$ and $P, M, Q_1 \ldots Q_m \in SN$ with $Sol(P \lessdot_{\mathcal{V}} M) = \{\varphi_1, \ldots, \varphi_n\}$. We claim that also $(\lambda_{\mathcal{V}} P.N)MQ_1 \ldots Q_m \in SN$. Indeed, reduction inside the terms P, M, N, $Q_1, \ldots, Q_m$ must terminate, since these terms are in SN by assumption. Hence, in finitely many steps, reducing the terms in $(\lambda_{\mathcal{V}} P.N)MQ_1 \ldots Q_m \in SN$ we obtain $(\lambda_{\mathcal{V}} P'.N')M'Q'_1 \ldots Q'_m \in SN$. Furthermore, its contraction gives $\sum_{i=1}^{r} N'\varphi'_i Q'_1, \ldots, Q'_m$ with $Sol(P' \lessdot_{\mathcal{V}} M') = \{\varphi'_1, \ldots, \varphi'_r\}$, which is a reduct of $\sum_{i=1}^{n} N\varphi_i Q_1 \ldots Q_m \in SN$. Since this term is in SN, the terms $\sum_{i=1}^{r} N'\varphi'_i Q'_1 \ldots Q'_m$ and $(\lambda_{\mathcal{V}} P.N)MQ_1 \ldots Q_m$ are in SN as well.

2. Suppose $S_1, S_2 \in SAT$, then we have to show $\{W \in \mathcal{T} \mid$ for all $Q \in S_1, WQ \in S_2\}$. Since S_1, S_2 are saturated then we have $WQ \in SN$ and $Q \in SN$ therefore $W \in SN$. So, indeed $\{W \mid$ for all $Q \in S_1, WQ \in S_2\} \subseteq SN$. Further, for l, for all $Q_1, \ldots, Q_m \in SN$ and for all $Q \in S_1$, $lQ_1 \ldots Q_m Q \in S_2$, because $Q \in SN$ and so $lQ_1 \ldots Q_m \in \{W \mid$ for all $Q \in S_1, WQ \in S_2\} \in SAT$. Finally, for all $m \geqslant 0$ and for all $Q_1, \ldots, Q_m, N, P \in SN$ with $\sum_{i=1}^{n} M\varphi_i Q_1 \ldots Q_m \in \{W \mid$ for all $Q \in S_1, WQ \in S_2\}$ where $Sol(P \lessdot_{\mathcal{V}} N) = \{\varphi_1, \ldots, \varphi_n\}$ we have to show $(\lambda_{\mathcal{V}} P.M)NQ_1 \ldots Q_m \in \{W \mid$ for all $Q \in S_1, WQ \in S_2\}$. We know that $\sum_{i=1}^{n} M\varphi_i Q_1 \ldots Q_m Q \in S_1 \Rightarrow (\lambda_{\mathcal{V}} P.M)NQ_1 \ldots Q_m Q \in S_2$ and so $(\lambda_{\mathcal{V}} P.M)NQ_1 \ldots Q_m \in \{W \mid$ for all $Q \in S_1, WQ \in S_2\}$.

3. Suppose $S_1, S_2 \in SAT$ and we have to show $\{W \mid$ for all $m \geqslant 0$ and for all $Q_1, \ldots, Q_m \in S_1, WQ_1 \ldots Q_m \in S_2\} \in SAT$. Since S_1, S_2 are saturated then we have $WQ_1 \ldots Q_m \in SN$ and $Q_1, \ldots, Q_m \in SN$ therefore $W \in SN$. So, indeed $\{W \mid$ for all $m \geqslant 0$ and for all $Q_1, \ldots, Q_m \in S_1, WQ_1 \ldots Q_m \in S_2\} \in SN$. Since S_2 is saturated for all $k, m \geqslant 0$ and for all $M_1, \ldots, M_k \in SN$ and for all $Q_1, \ldots, Q_m \in S_1$ we have $lM_1 \ldots M_k Q_1 \ldots Q_m \in S_2$ and so $lM_1 \ldots M_k \in \{W \mid$ for all $m \geqslant 0$ and for all $Q_1, \ldots, Q_m \in S_1, WQ_1 \ldots Q_m \in$

$S_2\} \in SN$. Finally, for all $k \geqslant 0$ and for all $M_1, \ldots, M_k, N, P \in SN$ with $\sum_{i=1}^{n} M\varphi_i M_1 \ldots M_k \in \{W \mid$ for all $m \geqslant 0$ and for all $Q_1, \ldots, Q_m \in S_1$, $WQ_1 \ldots Q_m \in S_2\}$ where $Sol(P \lessdot_{\mathcal{V}} N) = \{\varphi_1, \ldots, \varphi_n\}$ we have to show $(\lambda_{\mathcal{V}} P.M)NM_1 \ldots M_k \in \{W \mid$ for all $m \geqslant 0$ and for all $Q_1, \ldots, Q_m \in S_1$, $WQ_1 \ldots Q_m \in S_2\}$. We know that $\sum_{i=1}^{n} M\varphi_i M_1 \ldots M_k Q_1 \ldots Q_m \in S_2 \Rightarrow (\lambda_{\mathcal{V}} P.M)NM_1 \ldots M_k Q_1 \ldots Q_m \in B$ and so $(\lambda_{\mathcal{V}} P.M)NM_1 \ldots M_k \in \{W \mid$ for all $m \geqslant 0$ and for all $Q_1, \ldots, Q_m \in S_1, WQ_1 \ldots Q_n \in S_2\}$.

4. Suppose $\{S_i\}_{i \in I} \in SAT$, then by the definition of saturated sets for any $m \geqslant 0$ and for any $Q_1, \ldots, Q_m \in SN$ we have $xQ_1 \ldots Q_m \in S_i$ and, so $xQ_1 \ldots Q_m \in \cap_{i \in I} S_i$. Finally, for any $m \geqslant 0$ and for any $Q_1, \ldots, Q_m, N, P \in SN$ with $\sum_{j=1}^{n} M\varphi_j Q_1 \ldots Q_m \in S_i$ where $Sol(P \lessdot_{\mathcal{V}} N) = \{\varphi_1, \ldots, \varphi_n\}$ we have to show $(\lambda_{\mathcal{V}} P.M)NQ_1 \ldots Q_m \in \cap_{i \in I} S_i$. We know that $\sum_{j=1}^{n} M\varphi_j Q_1 \ldots Q_m \in S_i \Rightarrow (\lambda_{\mathcal{V}} P.M)NQ_1 \ldots Q_m \in S_i$ and so $(\lambda_{\mathcal{V}} P.M)NQ_1 \ldots Q_m \in \cap_{i \in I} S_i$.
5. We prove it by induction on the generation of τ.
   (a) $\tau = \alpha$ then by the definition of interpretation of a type and (1) of Lemma 6 we immediately have $[\![\alpha]\!] \in SAT$.
   (b) $\tau = \tau_1 \to \tau_2$ then by the definition of interpretation of a type we have $[\![\tau_1 \to \tau_2]\!] = \{W \in \mathcal{T} \mid$ for all $Q \in [\![\tau_1]\!], WQ \in [\![\tau_2]\!]\}$. Assume $[\![\tau_1]\!] \in SAT$ and $[\![\tau_2]\!] \in SAT$ then by (2) of Lemma 6 $[\![\tau_1 \to \tau_2]\!] \in SAT$.
   (c) $\tau = \tau_1^* \to \tau_2$ then by the definition of interpretation of a type we have $[\![\tau_1^* \to \tau_2]\!] = \{W \mid$ for all $m \geqslant 0$ and for all $Q_1, \ldots, Q_m \in [\![\tau_1]\!]$, $WQ_1 \ldots Q_m \in [\![\tau_2]\!]\}$. Assume $[\![\tau_1]\!] \in SAT$ and $[\![\tau_2]\!] \in SAT$ then by (3) of Lemma 6 $[\![\tau_1^* \to \tau_2]\!] \in SAT$.

□

Lemma 7. *If $W \in [\![\tau_1^* \to \tau_2]\!]$, then $WQ_1 \ldots Q_n \in [\![\tau_1^* \to \tau_2]\!]$ for all $n \geqslant 0$ and for all $Q_1, \ldots, Q_n \in [\![\tau_1]\!]$.*

Proof. Assume $W \in [\![\tau_1^* \to \tau_2]\!]$. By the definition of interpretation of a type we have $[\![\tau_1^* \to \tau_2]\!] = \{W \mid$ for all $n \geqslant 0$ and for all $Q_1, \ldots, Q_n \in [\![\tau_1]\!], WQ_1 \ldots Q_n \in [\![\tau_2]\!]\}$ and so we have $WQ_1 \ldots Q_n \in [\![\tau_1^* \to \tau_2]\!]$ for all $n \geqslant 0, Q_1, \ldots, Q_n \in [\![\tau_1]\!]$.

□

The notion of *satisfiability* is defined for substitutions and bases: Let φ be a substitution and Γ be a basis. Then

- φ satisfies $s : \rho$, written $\varphi \models s : \rho$, if $s\varphi \in [\![\rho]\!]$.
- φ satisfies Γ, written $\varphi \models \Gamma$, if $\varphi \models v : \rho$ for all $(v : \rho) \in \Gamma$.
- Γ satisfies $s : \rho$, written $\Gamma \models s : \rho$, if $\varphi \models \Gamma$ implies $\varphi \models s : \rho$ for all φ.

Lemma 8 (Soundness of Subtype Ordering). *If $\rho_1 \leqslant \rho_2$, then $[\![\rho_1]\!] \subseteq [\![\rho_2]\!]$.*

Proof. We prove by induction on the derivation of subtyping:

1. $\rho = \tau^* \to \tau' \leqslant \rho' = \tau^* \to \tau^* \to \tau'$. By the definition of interpretation of a type $[\![\tau^* \to \tau^* \to \tau']\!] = \{W \mid$ for all $m \geqslant 0$ and for all $Q_1, \ldots, Q_m \in [\![\tau]\!]$, $WQ_1 \ldots Q_m \in [\![\tau^* \to \tau']\!]\}$ and by Lemma 7 we can conclude $[\![\rho]\!] \subseteq [\![\rho']\!]$.

2. $\rho = \tau^* \to \tau' \leqslant \rho' = \tau'$. By the definition of interpretation of a type $[\![\tau^* \to \tau']\!] = \{W \mid \text{for all } m \geqslant 0 \text{ and for all } Q_1, \ldots, Q_m \in [\![\tau]\!], WQ_1 \ldots Q_m \in [\![\tau']\!]\}$ and by Lemma 7 we can conclude conclude $[\![\rho]\!] \subseteq [\![\rho']\!]$.
3. $\rho = \tau \leqslant \rho' = \tau^*$. By the definition of interpretation of a type $[\![\tau^*]\!] = \{(Q_1, \ldots, Q_m) \mid \text{for all } m \geqslant 0 \text{ and for all } Q_1, \ldots, Q_m \in [\![\tau]\!]\}$ and we can conclude $[\![\rho]\!] \subseteq [\![\rho']\!]$ (take $m = 1$).
4. $\rho = \tau_1^* \leqslant \rho' = \tau_2^*$ if $\tau_1 \leqslant \tau_2$. By the definition of interpretation of a type $[\![\tau_1^*]\!] = \{(Q_1, \ldots, Q_m) \mid \text{for all } m \geqslant 0 \text{ and for all } Q_1, \ldots, Q_m \in [\![\tau_1]\!]\}$ and $[\![\tau_2^*]\!] = \{(Q_1, \ldots, Q_m) \mid \text{for all } m \geqslant 0 \text{ and for all } Q_1, \ldots, Q_m \in [\![\tau_2]\!]\}$. By induction hypothesis we have $[\![\tau_1]\!] \subseteq [\![\tau_2]\!]$ and so if $W \in [\![\tau_1]\!]$ then $W \in [\![\tau_2]\!]$. So, if $(Q_1, \ldots, Q_m) \in [\![\tau_1^*]\!]$ we have $(Q_1, \ldots, Q_m) \in [\![\tau_2^*]\!]$.
5. $\rho = \tau_1 \to \tau_2 \leqslant \rho' = \tau_3 \to \tau_4$ if $\tau_3 \leqslant \tau_1$ and $\tau_2 \leqslant \tau_4$. By the definition of interpretation of a type $[\![\rho]\!] = [\![\tau_1 \to \tau_2]\!] = \{W \in \mathcal{T} \mid \text{for all } Q \in [\![\tau_1]\!], WQ \in [\![\tau_2]\!]\}$ and $[\![\rho]\!] = [\![\tau_3 \to \tau_4]\!] = \{W \in \mathcal{T} \mid \text{for all } Q \in [\![\tau_3]\!], WQ \in [\![\tau_4]\!]\}$. By induction hypothesis $[\![\tau_2]\!] \subseteq [\![\tau_4]\!]$ and $[\![\tau_3]\!] \subseteq [\![\tau_1]\!]$ it follows $[\![\tau_1 \to \tau_2]\!] \subseteq [\![\tau_3 \to \tau_4]\!]$.
6. $\rho = \tau_1^* \to \tau_2 \leqslant \rho' = \tau_3 \to \tau_4$ if $\tau_3 \leqslant \tau_1^*$ and $\tau_2 \leqslant \tau_4$. By the definition of interpretation of a type $[\![\tau_1^* \to \tau_2]\!] = \{W \mid \text{for all } m \geqslant 0 \text{ and for all } Q_1, \ldots, Q_m \in [\![\tau_1]\!], WQ_1 \ldots Q_m \in [\![\tau_2]\!]\}$ and $[\![\tau_3 \to \tau_4]\!] = \{W \mid \text{ for all } Q \in [\![\tau_3]\!], WQ \in [\![\tau_4]\!]\}$. By induction hypothesis $[\![\tau_2]\!] \subseteq [\![\tau_4]\!]$ and $[\![\tau_3]\!] \subseteq [\![\tau_1^*]\!]$ it follows $[\![\tau_1 \to \tau_2]\!] \subseteq [\![\tau_3 \to \tau_4]\!]$ (take $m = 1$).
7. $\rho = \tau_1^* \to \tau_2 \leqslant \rho' = \tau_3^* \to \tau_4$ if $\tau_3^* \leqslant \tau_1^*$ and $\tau_2 \leqslant \tau_4$. By the definition of interpretation of a type $[\![\tau_1^* \to \tau_2]\!] = \{W \mid \text{for all } n \geqslant 0 \text{ and for all } Q_1, \ldots, Q_n \in [\![\tau_1]\!], WQ_1 \ldots Q_n \in [\![\tau_2]\!]\}$ and $[\![\tau_3^* \to \tau_4]\!] = \{W \mid \text{for all } m \geqslant 0 \text{ and for all } Q_1, \ldots, Q_m \in [\![\tau_3]\!], WQ_1 \ldots Q_m \in [\![\tau_4]\!]\}$. By the induction hypothesis $[\![\tau_2]\!] \subseteq [\![\tau_4]\!]$ and $[\![\tau_3]\!] \subseteq [\![\tau_1]\!]$. So, we can conclude $[\![\tau_1^* \to \tau_2]\!] \subseteq [\![\tau_3^* \to \tau_4]\!]$.

□

The Soundness Theorem below states that if a statement can be derived from a basis, then the basis satisfies that statement.

Theorem 3 (Soundness). *If $\Gamma \vdash s : \rho$ then $\Gamma \models s : \rho$.*

Proof. By the structural induction on s:

1. If s is either x or X then trivially holds.
2. $\Gamma \vdash M : \tau$ with $M = M_1 M_2$ is a direct consequence of $\Gamma \vdash M_1 : \tau' \to \tau$ and $\Gamma \vdash M_2 : \tau'$. Suppose $\varphi \models \Gamma$ in order to show $\varphi \models M_1 M_2 : \tau$. Then $\varphi \models M_1 : \tau' \to \tau$ and $\varphi \models M_2 : \tau'$, i.e. $M_1\varphi \in [\![\tau' \to \tau]\!] = \{W \mid \text{ for all } Q \in [\![\tau']\!], WQ \in [\![\tau]\!]\}$ and $M_2\varphi \in [\![\tau']\!]$. But then by the definition of interpretation of a type $(M_1M_2)\varphi = M_1\varphi M_2\varphi \in [\![\tau]\!]$, i.e. $\varphi \models M_1 M_2 : \tau$.
3. $\Gamma \vdash M : \tau$ with $M = M_1 X$ is a direct consequence of $\Gamma \vdash M_1 : \tau'^* \to \tau$ and $\Gamma \vdash X : \tau'^*$. Suppose $\varphi \models \Gamma$ in order to show $\varphi \models M_1 X : \tau$. Then $\varphi \models M_1 : \tau'^* \to \tau$ and $\varphi \models X : \tau'^*$, i.e. $M_1\varphi \in [\![\tau'^* \to \tau]\!] = \{W \mid \text{for all } m \geqslant 0 \text{ and for all } Q_1, \ldots, Q_m \in [\![\tau']\!], WQ_1 \ldots Q_m \in [\![\tau]\!]\}$, and $M_2\varphi \in \{Q_1 \ldots Q_m \mid \text{for all } m \geqslant 0 \text{ and for all } Q_1, \ldots, Q_m \in [\![\tau']\!]\}$ But then by the definition of interpretation of a type $(M_1M_2)\varphi = M_1\varphi M_2\varphi \in [\![\tau]\!]$, i.e. $\varphi \models M_1 M_2 : \tau$.

4. $\Gamma \vdash M : \tau$ with $M = \lambda_{\mathcal{V}} M_1.M_2$ and $\tau = \tau_1 \to \tau_2$ is a direct consequence of $\Gamma \vdash M_1 : \tau_1$ and $\Gamma \vdash M_2 : \tau_2$. By the induction hypothesis we have $\Gamma \models M_1 : \tau_1$ and $\Gamma \models M_2 : \tau_2$. Suppose $\varphi \models \Gamma$ in order to show $\varphi \models \lambda_{\mathcal{V}} M_1.M_2 : \tau_1 \to \tau_2$. That is, we must show $\lambda_{\mathcal{V}} M_1.M_2\varphi N \in [\![\tau_2]\!]$ for all $N \in [\![\tau_1]\!]$. That is we must show $\sum_{i=1}^{n} M_2\varphi_i \in [\![\tau_2]\!]$ where $Sol(M_1 \lessdot_{\mathcal{V}} N) = \{\varphi_1, \ldots, \varphi_n\}$ and it holds according typing system and Lemma 5, i.e. $\varphi \models \lambda_{\mathcal{V}} M_1.M_2 : \tau$.
5. $\Gamma \vdash M : \tau$ with $M = M_1 + M_2$ is a direct consequence of $\Gamma \vdash M_1 : \tau$ and $\Gamma \vdash M_2 : \tau$. Suppose $\varphi \models \Gamma$ in order to show $\varphi \models M_1 + M_2 : \tau$. Then $\varphi \models M_1 : \tau$ and $\varphi \models M_2 : \tau$, i.e. $[\![M_1]\!]_\varphi \in [\![\tau]\!]$ and $M_2\varphi \in [\![\tau]\!]$. So, $M_1 + M_2\varphi = (M_1\varphi + M_2\varphi) \in [\![\tau]\!]$, i.e. $\varphi \models M_1 + M_2 : \tau$.
6. $\Gamma \vdash M : \tau_2$ is a direct consequence of $\Gamma \vdash M : \tau_1$ and $\tau_1 \leqslant \tau_2$. By induction hypothesis we have $\Gamma \models M : \tau_1$. Suppose $\varphi \models \Gamma$ in order to show $\varphi \models M : \tau_2$. Since $\Gamma \models M : \tau_1$, we have $\varphi \models M : \tau_1$, i.e., $M\varphi \in [\![\tau_1]\!]$. By Lemma 8 $[\![\tau_1]\!] \subseteq [\![\tau_2]\!]$, hence $M\varphi \in [\![\tau_2]\!]$, which shows that $\varphi \models M : \tau_2$.

□

These results imply Strong Normalization of the calculus:

Theorem 4 (Strong Normalization). *If $\Gamma \vdash s : \rho$, then $s \in SN$.*

Proof. Similar to the proof of the analogous theorem from [31].

Example 7. If variables were permitted to have arbitrary types instead of simple or starred simple types, then the SN theorem would not hold: Assume $(x : \tau^* \to \tau) \in \Gamma$ and $type(a) = \tau$. Then we can type the term $(\lambda x.(x\,x)\,\lambda x.(x\,x))$ as $\Gamma \vdash (\lambda x.(x\,x)\,\lambda x.(x\,x)) : \tau$. But its reduction does not terminate.

4 Application and Final Remarks

Jay in [19] underlines flexibility of pattern calculi, saying that "by making the class of patterns sufficiently generous, all the main programming styles can be expressed within a single, small calculus." In [17], the authors illustrate the power of patterns on the examples of XML manipulation with a general-purpose language bondi.

In this section we would like to follow the same approach, illustrating the power of patterns with hedge variables on the example of a rule-based language PρLog [14]. It is designed to perform conditional transformations on hedges, which may contain term, hedge, function, and context variables. Function and context variables are second-order variables of the type $\sigma^* \to \sigma$ and $\sigma \to \sigma$, respectively. Pattern matching should instantiate function variables with terms of the form $\lambda x_1, \ldots, x_n.h(x_1, \ldots, x_n)$, where h is a function symbol or a function variable. Context variables are instantiated by contexts: terms of the form $\lambda x.M$, where x occurs in M freely exactly once [36]. Its rules are built from atoms $rule(strategy(\langle h_1 \rangle), \langle h_2 \rangle)$, usually written as $strategy :: h_1 \Rightarrow h_2$. The left hand side is a pattern. Pattern matching will use the corresponding matching algorithm from [25].

PρLog is built on top of Prolog and inherits its evaluation strategy and built-in primitives. It does not have explicit typing, but one can restrict possible instantiations of variables by regular (hedge or context) languages, specifying the corresponding constraint. It can use Prolog clauses in its code. We are not going to describe PρLog in detail here. The interested reader can find the description of the main features in [14]. The code is freely available from `http://www.risc.jku.at/people/tkutsia/software.html`. Here we are interested in the illustration of its capabilities, taking advantage of the flexibility the patterns provide. This we do for an XML query used for the same purposes in [17], emphasizing how the pattern structures can be declared. Hedge variables, higher-order variables, and unranked symbols come particularly handy.

Consider geographical data of the form `data(country*)`, consisting of finite (possibly empty) sequence of countries. Each country is represented as `country(name,area(n),pop(m),province*)`, where `name` is the name of the country, `area(n)` is the area, `pop(m)` is the population. Each province is represented as `province(name,area(n),pop(m),city*)` and each city is represented as `city(name, pop(m),rivers(rname*))`, where `rname*` is the sequence of river names running through the city. The relative order of the area and population terms in not fixed, i.e, we may have `province(...,area(n),pop(m),...)` as well as `province(...,pop(m), area(n),...)`.

We may want to extract a sequence of names of cities whose population is bigger than 300 (in units of thousands). Instead of writing PρLog clauses for this particular query, we can write a more generic one, which can be used for extracting also province names whose population is greater or less than a given number, or whose area is greater or less than a given number, and the same for the countries. The code consists of the following three clauses:

```
extract_all(f_dist, f_pa, i_test) ::
        c_X(f_dist(i_n,s_,f_pa(i_x),s_)) ==> (i_n,s_names) :-
    i_test :: i_x ==> true,
    !,
    extract_all(f_dist, f_pa, i_test) :: c_X(f_dist) ==> s_names.
extract_all(f_dist, f_pa, i_test) ::
        c_X(f_dist(i_n,s_,f_pa(i_x),s_)) ==> s_names :-
    !,
    extract_all(f_dist, f_pa, i_test) :: c_X(f_dist) ==> s_names.
extract_all(f_dis, f_pa, i_test) :: i_ ==> eps.
```

The term variables (individual variables) start with `i_`, hedge variables (sequence variables) with `s_`, function variables with `f_`, and context variables with `c_`. As one can see, the strategy name `extract_all(f_dist, f_pa, i_test)` is parametrized with the constructor of the territorial district (`f_dist` may stand for the constructors `country`, `province`, or `city`), with the constructor standing for population or area (`f_pa` may become `pop` or `area`), and by the comparison test (`i_test` can be instantiated, for example, by `greater(300)`, which means that what we are looking for, area or population, should be greater than 300 in

the corresponding units). The left hand side of the first rule `c_X(f_dist(i_name, s_,f_pa(i_x),s_))` is the pattern. It should match the data term, to select the name `i_n` and the number `i_x` at arbitrary depth under the constructors `f_dist` and `f_pa`, respectively, ignoring the term sequences before and after `f_pa(i_x)`. (The anonymous variable `s_` matches any hedge.) The hedge `(i_n,s_names)` in the right hand side is the answer, provided that the test `i_test` succeeds for `i_x` (the program is supposed to have the rules for this test), and `s_names` is the hedge of names extracted from the other parts of the data. These two conditions are in the body of the clause. The `!` is the standard Prolog cut predicate.

Let the concrete data term be denoted by D. It contains information about area, population, provinces, cities, and rivers, in certain countries. If we get back to our original goal, extracting a sequence of names of cities whose population is bigger than 300 thousand, we can formulate it simply as `extract_all(city,pop, greater(300)) :: D ==> s_X`. To extract a sequence of countries whose area is smaller than 200 thousand square km, we can write `extract_all(country, area,smaller(200)) :: D ==> s_X`, etc. This example shows how path and pattern polymorphisms in the sense of [19] can be expressed in a language based on pattern calculus with hedge variables.

We believe that a notion of definability in our calculus may lead to the basis of the construction of efficient compilers for a functional language extended with hedge variables. In fact it is possible to represent data types with a flexible number of elements in a very direct manner in our calculus. With this in mind our next step is the definition of a core functional programming language with hedge variables and its encoding in the calculus presented here.

Acknowledgements

Partially supported by the FCT fellowship (ref. SFRH/BD/62058/2009), by the Austrian Science Fund (FWF) under the project SToUT (P 24087-N18), and by the Rustaveli Science Foundation under the grant DI/16/4-120/11.

References

1. S. Anantharaman, P. Narendran, and M. Rusinowitch. Unification modulo *acui* plus distributivity axioms. *J. Autom. Reasoning*, 33(1):1–28, 2004.
2. A. Arbiser, A. Miquel, and A. Ríos. The lambda-calculus with constructors: Syntax, confluence and separation. *J. Funct. Program.*, 19(5):581–631, 2009.
3. P. Arrighi and G. Dowek. Linear-algebraic lambda-calculus: higher-order, encodings, and confluence. In Voronkov [37], pages 17–31.
4. H. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. North-Holland, 1984. Revised edition.
5. G. Barthe, H. Cirstea, C. Kirchner, and L. Liquori. Pure patterns type systems. In A. Aiken and G. Morrisett, editors, *POPL*, pages 250–261. ACM, 2003.
6. V. Benzaken, G. Castagna, and A. Frisch. CDuce: an XML-centric general-purpose language. In C. Runciman and O. Shivers, editors, *ICFP*, pages 51–63. ACM, 2003.

7. H. Boley. *A Tight, Practical Integration of Relations and Functions*, volume 1712 of *LNCS*. Springer, 1999.
8. E. Chasseur and Y. Deville. Logic program schemas, constraints, and semi-unification. In N. E. Fuchs, editor, *LOPSTR*, volume 1463 of *LNCS*, pages 69–89. Springer, 1997.
9. H. Cirstea and G. Faure. Confluence of pattern-based calculi. In F. Baader, editor, *RTA*, volume 4533 of *LNCS*, pages 78–92. Springer, 2007.
10. H. Cirstea and C. Kirchner. The rewriting calculus - parts I and II. *Logic Journal of the IGPL*, 9(3), 2001.
11. J. Coelho, B. Dundua, M. Florido, and T. Kutsia. A rule-based approach to XML processing and Web reasoning. In P. Hitzler and T. Lukasiewicz, editors, *RR*, volume 6333 of *LNCS*, pages 164–172. Springer, 2010.
12. J. Coelho and M. Florido. CLP(Flex): Constraint Logic Programming applied to XML processing. In R. Meersman and Z. Tari, editors, *CoopIS/DOA/ODBASE (2)*, volume 3291 of *LNCS*, pages 1098–1112. Springer, 2004.
13. A. Díaz-Caro and B. Petit. Linearity in the non-deterministic call-by-value setting. In C.-H. L. Ong and R. J. G. B. de Queiroz, editors, *WoLLIC*, volume 7456 of *LNCS*, pages 216–231. Springer, 2012.
14. B. Dundua, T. Kutsia, and M. Marin. Strategies in Pρlog. In M. Fernández, editor, *WRS*, volume 15 of *EPTCS*, pages 32–43, 2009.
15. T. Ehrhard and L. Regnier. The differential lambda-calculus. *Theor. Comput. Sci.*, 309(1-3):1–41, 2003.
16. H. Hosoya and B. C. Pierce. Regular expression pattern matching for XML. *J. Funct. Program.*, 13(6):961–1004, 2003.
17. F. Y. Huang, B. Jay, and D. Skillicorn. Programming with heterogeneous structures: Manipulating XML data using bondi. In V. Estivill-Castro and G. Dobbie, editors, *ACSC*, volume 48 of *CRPIT*, pages 287–296. Australian Comp. Soc., 2006.
18. F. Jacquemard and M. Rusinowitch. Closure of hedge-automata languages by hedge rewriting. In Voronkov [37], pages 157–171.
19. B. Jay. *Pattern Calculus*. Springer, 2009.
20. C. B. Jay and D. Kesner. Pure pattern calculus. In P. Sestoft, editor, *ESOP*, volume 3924 of *LNCS*, pages 100–114. Springer, 2006.
21. T. Kutsia. Theorem proving with sequence variables and flexible arity symbols. In M. Baaz and A. Voronkov, editors, *LPAR*, volume 2514 of *LNCS*, pages 278–291. Springer, 2002.
22. T. Kutsia. Solving equations with sequence variables and sequence functions. *J. Symb. Comput.*, 42(3):352–388, 2007.
23. T. Kutsia. Flat matching. *J. Symb. Comput.*, 43(12):858–873, 2008.
24. T. Kutsia and B. Buchberger. Predicate logic with sequence variables and sequence function symbols. In A. Asperti, G. Bancerek, and A. Trybulec, editors, *MKM*, volume 3119 of *LNCS*, pages 205–219. Springer, 2004.
25. T. Kutsia and M. Marin. Matching with regular constraints. In G. Sutcliffe and A. Voronkov, editors, *LPAR*, volume 3835 of *LNCS*, pages 215–229. Springer, 2005.
26. T. Kutsia and M. Marin. Solving, reasoning, and programming in Common Logic. In *Proc. 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2012*. IEEE Computer Society, 2012. To appear. The preliminary version is available from http://www.risc.jku.at/people/tkutsia/papers/KutsiaMarin2012CL.pdf.
27. C. Menzel. Knowledge representation, the world wide web, and the evolution of logic. *Synthese*, 182(2):269–295, 2011.

28. J. Meseguer and J. A. Goguen. Order-sorted unification. *J. Symb. Comput.*, 8(4):383–413, 1989.
29. M. Pagani and S. R. D. Rocca. Solvability in resource lambda-calculus. In C.-H. L. Ong, editor, *FOSSACS*, volume 6014 of *LNCS*, pages 358–373. Springer, 2010.
30. B. Petit. Semantics of typed lambda-calculus with constructors. *Logical Methods in Computer Science*, 7(1), 2011.
31. J. Rehof. Strong normalization for non-structural subtyping via saturated sets. *Inf. Process. Lett.*, 58(4):157–162, 1996.
32. M. Schmidt-Schauß. *Computational Aspects of an Order-Sorted Logic with Term Declarations*, volume 395 of *LNCS*. Springer, 1989.
33. G. Smolka, W. Nutt, J. A. Goguen, and J. Meseguer. Order-sorted equational computation. In M. Nivat and H. Aït-Kaci, editors, *Resolution of Equations in Algebraic Structures*, volume 2, pages 297–367. Academic Press, 1989.
34. M. Sulzmann and K. Z. M. Lu. Xhaskell - adding regular expression types to haskell. In O. Chitil, Z. Horváth, and V. Zsók, editors, *IFL*, volume 5083 of *LNCS*, pages 75–92. Springer, 2007.
35. V. van Oostrom. Lambda calculus with patterns. Technical Report IR-228, Vrije Universiteit, Amsterdam, 1990.
36. M. Villaret. *On Some Variants of Second-Order Unification*. PhD thesis, Technical University of Catalonia, Barcelona, 2004.
37. A. Voronkov, editor. *Rewriting Techniques and Applications, 19th International Conference, RTA 2008, Hagenberg, Austria, July 15-17, 2008, Proceedings*, volume 5117 of *LNCS*. Springer, 2008.
38. B. Wack and C. Houtmann. Strong normalisation in two pure pattern type systems. *Mathematical Structures in Computer Science*, 18(3):431–465, 2008.
39. M. Wand. Complete type inference for simple objects. In *LICS*, pages 37–44. IEEE Computer Society, 1987.
40. S. Wolfram. *The Mathematica Book*. Wolfram Media, 5th edition, 2003.