Normalisation for Dynamic Pattern Calculi

Eduardo Bonelli[1,2], Delia Kesner[3], Carlos Lombardi[1,3,4], and Alejandro Ríos[4]

1 Univ. Nac. de Quilmes, Argentina
2 CONICET, Argentina
3 Univ. Paris Diderot, Sorbonne Paris Cité, PPS, CNRS, France
4 Univ. de Buenos Aires, Argentina

Abstract

The Pure Pattern Calculus (PPC) [10, 11] extends the λ-calculus, as well as the family of algebraic pattern calculi [20, 6, 12], with first-class patterns *i.e.* patterns can be passed as arguments, evaluated and returned as results. The notion of *matching failure* of PPC in [11] not only provides a mechanism to define functions by pattern matching on cases but also supplies PPC with parallel-or-like, non-sequential behaviour. Therefore, devising normalising strategies for PPC to obtain well-behaved implementations turns out to be challenging.

This paper focuses on normalising reduction strategies for PPC. We define a (multistep) strategy and show that it is normalising. The strategy generalises the leftmost-outermost strategy for λ-calculus and is strictly finer than parallel-outermost. The normalisation proof is based on the notion of *necessary set of redexes*, a generalisation of the notion of needed redex encompassing non-sequential reduction systems.

1998 ACM Subject Classification F.4.1 Mathematical Logic, D.3.3 Language Constructs and Features

Keywords and phrases Pattern calculi, reduction strategies, sequentiality, neededness

Digital Object Identifier 10.4230/LIPIcs.RTA.2012.117

Category Regular Research Paper

1 Introduction

Pattern calculi [20, 12, 6, 15] model the pattern-matching primitives of functional programming languages (*e.g.* OCAML, ML, Haskell) and proof assistants (*e.g.* Coq, Isabelle), where functions can be defined by n different *cases* by writing for example $\mathtt{f} := p_1\texttt{->}s_1 \mid \ldots \mid p_n\texttt{->}s_n$. When p_i is typically expressed in terms of *constructors* and variables we speak of *algebraic pattern calculi*. The application of f to an argument u starts by matching p_1, the *pattern* of the first case, against u. If such a matching is successful, then it yields a substitution σ_1 whose domain is the set of variables in p_1. This substitution is applied to s_1, the *body* of the first case, and then the substituted body is evaluated. If a successful matching is not possible, *i.e.* there is a *matching failure*, then evaluation continues with the following cases in the definition of f.

The Pure Pattern Calculus (PPC). PPC [10, 11] generalises algebraic pattern calculi (and hence the λ-calculus) by allowing an *arbitrary* term t_i to take the role of a pattern p_i thus generalising a function to $\mathtt{f} := t_1\texttt{->}s_1 \mid \ldots \mid t_n\texttt{->}s_n$. Reduction inside the t_i is now allowed, hence patterns may be computed *dynamically*. Also, symbols in t_i now play two roles: either they are *variables* (*i.e.* place holders) for terms (which substitution from "outside" will duly

23rd International Conference on Rewriting Techniques and Applications (RTA'12).
Editor: A. Tiwari; pp. 117–132
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

fill in) or they are *matchables* (depicted with a hat for easy identification) in the sense that they are only used to decompose data. These two roles are distinguished by decorating a case t`->`s with a set θ of *binding symbols* that singles out the subset of matchables in the pattern t that are meant for matching, the variables in the body s being those acting as place-holders. Consider for example `elim` $::= \lambda_{\{x\}}\ \widehat{x}.(\lambda_{\{y\}}\ x\ \widehat{y}.y)$ where we write $\lambda_\theta\ t.s$ rather than `t` $\texttt{->}_\theta$ `s`. The inner abstraction $\lambda_{\{y\}}\ x\ \widehat{y}.y$ binds the only occurrence of the *matchable* $\widehat{y}$ in the pattern $x\ \widehat{y}$ and that of the *variable* y in the body y; the x in $x\ \widehat{y}$ is excluded from $\{y\}$ since it acts as a place-holder in that pattern. However, the occurrence of x, as well as that of $\widehat{x}$, are both bound by the outermost $\lambda_{\{x\}}$, as can be seen in Fig. 1.

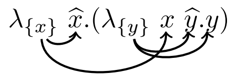


Figure 1

The dynamics of reduction in `PPC` may be illustrated by the reduction sequence of Fig. 2 in which `elim` is applied to the function $\lambda_{\{z\}}\ \widehat{z}.$`cons` z `nil`, where `cons` and `nil` denote constructors. In the first step, $\lambda_{\{z\}}\ \widehat{z}.$`cons` z `nil` is substituted for x into the pattern $x\ \widehat{y}$. In the second step, the resulting application $(\lambda_{\{z\}}\ \widehat{z}.\texttt{cons}\ z\ \texttt{nil})\,\widehat{y}$, which resides in a pattern, is reduced. The resulting term, when applied to an argument, will yield a succesful matching only if this argument is a *compound data* of the form `cons` t `nil`. This example exhibits the *pattern polymorphism* capabilities of `PPC` allowing to obtain structurally different deconstructors by applying the same term `elim` to different arguments.

$$
\begin{aligned}
&(\lambda_{\{x\}}\ \widehat{x}.(\lambda_{\{y\}}\ x\ \widehat{y}.y))\,(\lambda_{\{z\}}\ \widehat{z}.\texttt{cons}\ z\ \texttt{nil})\\
\to\ &\lambda_{\{y\}}\ (\lambda_{\{z\}}\ \widehat{z}.\texttt{cons}\ z\ \texttt{nil})\,\widehat{y}.y\\
\to\ &\lambda_{\{y\}}\ \texttt{cons}\ \widehat{y}\ \texttt{nil}.y
\end{aligned}
$$

Figure 2

Reduction in Fig. 2 proceeded smoothly. However, it may fail. In `PPC`, $(\lambda_\theta\ p.s)t$ is reducible only if the match of the pattern p against the argument t is *decided*, *i.e.* it is either a successful match, specified by a substitution, or a matching failure, specified by a closed *normal form* (written[1] ***nf***). *E.g.* consider $(\lambda_{\{x\}}\ \texttt{a}\ r_1.x)(\texttt{b}\ r_2)$ where r_1 and r_2 are *redexes*. Matching of `a` r_1 against `b` r_2 fails since the head constructors `a` and `b` are different. This is also the case for $(\lambda_{\{x\}}\ \texttt{a}\ r_1\ \texttt{b}.x)(\texttt{a}\ r_2\ \texttt{d})$, now because the constructors `b` and `d` mismatch, a fact that can be established without examining r_1 or r_2. Indeed, failure of matching for *any* component of a given compound data automatically implies failure of matching for the entire compound.

Non-sequentiality of `PPC`. What would be a smart *normalising* strategy for `PPC`? Clearly, if the matching of p against t is decided when evaluating $(\lambda_\theta p.s)t$, then it is a redex and should be selected. However, when it is non-decided, it is necessary to understand which subterm $(p, s$ or $t)$ needs to be evaluated first in order to attain a normal form for the whole term. *E.g.* it is the *pattern* that must be reduced in $(\lambda_{\{x\}}\ \texttt{I}\ (\texttt{a}\ \widehat{x}).x)(\texttt{a c})$ (`I` is the identity function) in order to attain a decided match, the *argument* in $(\lambda_{\{x\}}\ \texttt{a}\ \widehat{x}.x)(\texttt{I}\ (\texttt{a c}))$, and *both* the pattern and the argument in $(\lambda_{\{x\}}\ \texttt{a}\ (\texttt{I}\ (\texttt{b}\ \widehat{x})).x)((\texttt{I a})\ \texttt{b}\ y)$. Even though we could establish that the pattern or the argument or both have to be reduced, deciding *which* redex to pick in each of these cases is not always possible. An example is $t_1 := (\lambda_{\{x\}}\ \texttt{a}\ (\texttt{b}\ \widehat{x})\ r_1.r_2)\ (\texttt{a}\ r_3\ (\texttt{d}\ r_4))$ where matching is non-decided: both pattern and argument must be reduced in order for t_1 to become a redex. Reducing r_1 is not necessarily a good idea. Indeed, suppose r_1 is any non-terminating computation and r_3 reduces to `d` t, for some t. Then reduction of r_3 causes

[1] There are other different reasonable approaches.

a matching failure since b and d mismatch. We could rather have selected r_3, however if this time it is r_3 that is non-terminating and r_1 reduces to b t, for some t, then again we miss failure and loop. The same happens if we choose r_4. The term t_1 illustrates that PPC fails to be *sequential*.

Informally, sequentiality means that given a term of the form $C[r_1, \ldots, r_n]$ where C does not contain any redex and every r_i is a redex, there exists an index i s.t. r_i is a *needed* redex and the choice of i is independent from $r_1, \ldots, r_n$. A redex r in a term t is needed iff every reduction sequence from t to normal form reduces (a *residual* of) r. Another example showing terms not in normal form may not have needed redexes is $t_2 :=$ $(\lambda_{\{y\}}$ a b c $\widehat{y}.y)$ (a (I c) (I b) (I a)) (redexes are shown in gray). This term admits at least two different reduction sequences to normal form: $t_2 \to (\lambda_{\{y\}}$ a b c $\widehat{y}.y)$(a c (I b) (I a)) $\to$ ***nf*** and also $t_2 \to (\lambda_{\{y\}}$ a b c $\widehat{y}.y)$(a (I c) b (I a)) $\to$ ***nf***. Sequentiality fails in PPC because matching may fail for *different reasons*: none of the redexes in t_2 is needed since failure of matching can be declared in terms of (only) I b or I c.

Even if dynamic patterns and non-sequentiality are central issues of this paper, the calculus would also be non-sequential if the set of patterns were restricted to the static ones. As explained before, non-sequentiality comes from the notion of matching failure introduced in [11], which is detailed in Sec. 2, and applies to any kind of static pattern, including the standard, algebraic ones.

Towards normalisation strategies for PPC. It has been shown in different settings that repeated contraction of needed redexes yields normalising strategies for sequential rewriting systems. Failure of sequentiality leaves us with two avenues to pursue in order to devise normalising strategies for PPC. The first is to introduce *look-ahead* by performing several reduction steps in order to identify an i. In this case, apart from C we would have to examine the r_is. Such an approach is overly expensive since it involves testing for cycles [1]. The second avenue consists in selecting a *multistep*: a *set* of redexes reduced simultaneously at each step. Of particular appeal in this approach is the notion of *necessary set of redexes*: a set of redexes $\mathcal{A}$ in a term t is called necessary iff every reduction sequence from t to normal form reduces at least (a *residual* of) one element of $\mathcal{A}$, even if this element may differ from sequence to sequence. The set of all redexes, and also the set of all outermost redexes, are necessary sets for any term in PPC. However, implementations could benefit from *finer* strategies, *i.e.* reduction strategies selecting smaller sets of redexes in each multistep.

Contributions. We propose a strategy for PPC which selects for each term a necessary set bounded by the set of its outermost redexes and show that it is normalising. More precisely, we introduce:

- a theory of needed normalisation for PPC by adapting the notion of *necessary sets* [21] to the higher-order case;
- a proof that repeated contraction of *necessary sets* of redexes is normalising, provided that those sets also enjoy an additional property, namely they are *non-gripping* [16];
- an inductively formulated strategy for PPC that produces necessary, non-gripping sets of redexes, and hence is normalising.

Related work. In λ-calculus and, more generally, orthogonal higher-order (HO) Expression Reduction Systems, any normalising term not in normal form contains a needed redex and (one-step) contraction of needed redexes attains a normal form [5, 7]. Unfortunately, as discussed above, terms in PPC not in normal form may contain no needed redexes. Multistep reduction strategies have been studied for both first-order (FO) and HO rewriting. Normalising multistep strategies by means of necessary sets have been defined for non-sequential FO TRS in [21], which has been our main source of inspiration. Van Raamsdonk [23] extends O'Donnell's result [19] (reduction of all outermost redexes is normalising) to almost orthogonal HORS. Indeed, [23] proves that being *outermost fair* is a sufficient condition for a reduction strategy to be normalising; this result was then extended [22] to weakly orthogonal HO rewriting. These works have influenced some of the concepts and technical tools used in this paper. Melliès [16, 17] develops an axiomatic theory of neededness for (possibly) overlapping rewrite systems, motivated by the work of Huet and Lévy for orthogonal FO TRS [8]. Pattern calculi may be modeled as the axiomatic rewrite systems of Melliès, however his normalisation results are not applicable since pattern calculi do not enjoy *stability* [16](Axiom IV, pg.80): there may be more than one way to create the same redex due to matching failure.

Regarding the specific setting of non-sequential pattern calculi (including matching failure) we are not aware of any literature on normalising strategies. For an abridged version of `PPC` resulting from restricting the set of patterns to the algebraic ones and from disallowing matching to fail, one-step *standard* reductions (obtained by means of a standardisation theorem) turn out to be normalising [13]. Also worth mentioning is the existence of sequential (*i.e.* every term has a needed redex) operational semantics of dynamic pattern calculi appearing for example in [9, 3, 4]; they can be understood as (re)formulations of `PPC` where the conditions determining when a match should fail impose a more restricted evaluation order.

Structure of the paper. Sec. 2 introduces `PPC`. Sec. 3 defines multistep strategies and develops the tools needed to guarantee that complete developments can be used as multisteps. Sec. 4 presents a reduction strategy $\mathcal{S}$ for `PPC`. Sec. 5 formalises the notion of necessary sets of redexes, motivates and introduces the additional non-gripping property, shows that $\mathcal{S}$ always reduces necessary and non-gripping sets, and uses these facts to prove that it is normalising. Finally, Sec. 6 concludes and suggests further work.

2 The Pure Pattern Calculus

This section briefly introduces `PPC` following the presentation of [11][2].

Syntax: Consider a countable set of **symbols** $f, g, \ldots, x, y, z$. Sets of symbols are denoted by meta-variables θ, ϕ, ... The syntax of `PPC` is summarised by the following grammar:

Terms	$(\boldsymbol{T})$	t	$::=$	$x \mid \widehat{x} \mid tt \mid \lambda_\theta\ t.t$
Data-Structures	$(\boldsymbol{DS})$	D	$::=$	$\widehat{x} \mid Dt$
Matchable-forms	$(\boldsymbol{MF})$	F	$::=$	$D \mid \lambda_\theta\ t.t$

The term x is called a **variable**, $\widehat{x}$ a **matchable**, tu an **application** (t is the **function** and u the **argument**) and $\lambda_\theta\ p.u$ an **abstraction** (θ is the set of **binding symbols**, p is the **pattern** and u is the **body**). Application (resp. abstraction) is left (resp. right) associative. A λ-abstraction $\lambda x.t$ can be defined by $\lambda_{\{x\}}\ \widehat{x}.t$. The **identity function** $\lambda_{\{x\}}\ \widehat{x}.x$ is abbreviated `I`.

[2] Other presentations are [10, 9], but both of them use sequential, rather than parallel-or like, semantics.

A binding symbol $x \in \theta$ of an abstraction $\lambda_\theta\ p.s$ *binds* matchable occurrences of x in p and variable occurrences of x in s. The derived notions of **free variables** and **free matchables** are respectively denoted by $\texttt{fv}(\_)$ and $\texttt{fm}(\_)$. This is illustrated in Fig. 3.

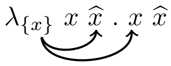


Figure 3

As usual, we consider terms up to **alpha-conversion**, *i.e.* up to renaming of bound matchables and variables. **Constructors** are matchables which are not bound and, to ease the presentation, they are often denoted in typewriter fonts $\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}, \ldots$, thus for example $\lambda_{\{x,y\}}\ \widehat{x}\ y\ \texttt{a}.y$ denotes $\lambda_{\{x,y\}}\ \widehat{x}\ y\ \widehat{z}.y$. The distinction between matchables and variables is unnecessary for standard (static) patterns which do not contain free variables.

A **position** is either ϵ (the empty position), or na, where $n \in \{1, 2\}$ and a is a position. We use $a, b, \ldots$ (resp. $\mathcal{A}, \mathcal{B}, \ldots$ and $\delta, \rho, \pi, \ldots$) to denote positions (resp. sets and sequences of positions) and $b\mathcal{A}$ to mean $\{ba \mid a \in \mathcal{A}\}$. The **set** $\texttt{Pos}(t)$ **of positions** of t is defined as expected, provided that for abstractions $\lambda_\theta\ p.s$ positions inside both p and s are considered. Here is an example $\texttt{Pos}(\lambda_{\{x\}}\ \texttt{a}\ \texttt{b}.\texttt{a}\ x\ x) = \{\epsilon, 1, 2, 11, 12, 21, 22, 211, 212\}$.

We write $t|_a$ for the **subterm of** t **at position** a and $t[s]_a$ for the **replacement** of the subterm at position a in t by s. Notice that replacement may capture variables. We write $a \leq b$ (resp. $a \parallel b$) when the position a is a **prefix** of (resp. **disjoint** from) the position b. All these notions are defined as expected [2] and extended to sets of positions as well.

Substitution and Matching: A **substitution** σ is a mapping from variables to terms with finite domain $\texttt{dom}(\sigma)$. A **match** μ is either a substitution or a special constant in the set $\{\texttt{fail}, \texttt{wait}\}$. A **match** is **positive** if it is a substitution; it is **decided** if it is either positive or $\texttt{fail}$. The notions of domain and free variables/matchables are naturally extended to matches, in particular, the domain of $\texttt{fail}$ is the empty set and that of $\texttt{wait}$ is undefined. The **application of a substitution** σ to a term is written and defined as usual on alpha-equivalence classes. The **application of a match** μ to a term t, written μt, is defined as follows: if μ is a substitution, then it is applied as explained above; if $\mu = \texttt{wait}$, then μt is undefined; if $\mu = \texttt{fail}$, then μt is the identity function $\texttt{I}$. Other *closed terms in normal form* could be taken to define the last case, this one allows in particular to encode pattern-matching definitions given by alternatives [11].

The **disjoint union** of two matches μ_1 and μ_2 is a crucial operation used to define the operational semantics of $\texttt{PPC}$. Disjoint union is written $\mu_1 \uplus \mu_2$ and is defined as: their union if both μ_i are substitutions and $\texttt{dom}(\mu_1) \cap \texttt{dom}(\mu_2) = \emptyset$; $\texttt{wait}$ if either of the μ_i is $\texttt{wait}$ and none is $\texttt{fail}$; $\texttt{fail}$ otherwise. This definition of disjoint union of matches validates the following equations which are responsible for the non-sequential nature of $\texttt{PPC}$:

$$\texttt{fail} \uplus \texttt{wait} = \texttt{wait} \uplus \texttt{fail} = \texttt{fail}$$

We return to these equations immediately after the definition of the operational semantics of the calculus. The **compound matching operation** takes a term, a set of binding symbols and a pattern and returns a match, it is defined by applying the following equations in order:

$$\begin{array}{rcll}
\{\!\{\widehat{x} \rhd_\theta t\}\!\} & := & \{x \to t\} & \text{if } x \in \theta \\
\{\!\{\widehat{x} \rhd_\theta \widehat{x}\}\!\} & := & \{\} & \text{if } x \notin \theta \\
\{\!\{pq \rhd_\theta tu\}\!\} & := & \{\!\{p \rhd_\theta t\}\!\} \uplus \{\!\{q \rhd_\theta u\}\!\} & \text{if } tu, pq \in \boldsymbol{MF} \\
\{\!\{p \rhd_\theta t\}\!\} & := & \texttt{fail} & \text{if } p, t \in \boldsymbol{MF} \\
\{\!\{p \rhd_\theta t\}\!\} & := & \texttt{wait} & \text{otherwise}
\end{array}$$

The use of disjoint union in the third case of the previous definition restricts compound matching to linear patterns [3], which is necessary to guarantee confluence. Indeed, disjoint union of two substitutions fails whenever their domains are not disjoint.

The result of the **matching operation**[4] $\{p/_\theta\ t\}$ is defined to be the *check* of $\{\!\{p \rhd_\theta t\}\!\}$ on θ; where the **check** of a match μ on θ is `fail` if μ is a substitution whose domain is not θ, μ otherwise. Notice that $\{p/_\theta\ t\}$ is never positive if p is not linear with respect to θ. We now give some examples: $\{\widehat{x}\widehat{x}/_{\{x\}}\ uv\}$ gives `fail` because $\widehat{x}\widehat{x}$ is not linear; $\{\widehat{x}\widehat{y}/_{\{x,y,z\}}\ uv\}$ gives `fail` because $\{x,y,z\} \neq \{x,y\}$, $\{\widehat{x}/_\emptyset\ u\}$ gives `fail` because $\emptyset \neq \{x\}$; $\{\widehat{y}/_{\{x\}}\ \widehat{y}\}$ gives `fail` because $\{x\} \neq \emptyset$; $\{\widehat{x}\widehat{y}/_{\{x\}}\ u\widehat{z}\}$ gives `fail` because $\{\!\{\widehat{y} \rhd_{\{x\}} \widehat{z}\}\!\}$ is `fail`; $\{\widehat{x}\widehat{y}/_\emptyset\ u\widehat{z}\}$ gives `fail` for the same reason.

Semantics: The semantics of `PPC` is given by means of the following **reduction rule**:

$$(\lambda_\theta\ p.s)u \mapsto \{p/_\theta\ u\}s, \text{ if } \{p/_\theta\ u\} \text{ is decided}$$

The rule applies to the term $(\lambda_{\{x,y\}}\ \mathtt{a}\ \widehat{x}\ \widehat{y}.y\ x)(\mathtt{a}\ \mathtt{b}\ (\mathtt{I}\ \mathtt{a}))$ yielding $(\mathtt{I}\ \mathtt{a})\ \mathtt{b}$; and also to $(\lambda_{\{x,y\}}\ \mathtt{a}\ \widehat{x}\ \widehat{y}.y\ x)(\mathtt{c}\ \mathtt{b}\ (\mathtt{I}\ \mathtt{a}))$ yielding `I` (since the matching operation yields `fail`). However, the rule does not apply to the term $(\lambda_{\{x,y\}}\ \mathtt{a}\ \widehat{x}\ \widehat{y}.y\ x)(\mathtt{I}\ \widehat{x})$ since $\{\mathtt{a}\ \widehat{x}\ \widehat{y}/_{\{x,y\}}\ \mathtt{I}\ \widehat{x}\}$ is `wait`.

It is worth noticing that sequentiality of `PPC` can be recovered (see *e.g.* [9, 3, 4]) by modifying the equations of disjoint union, however, some meaningful terms will no longer be normalising. Thus for example, if $\mathtt{fail} \uplus \mu$ is defined to be `fail`, while $\mathtt{wait} \uplus \mathtt{fail} = \mathtt{wait}$ and $\sigma \uplus \mathtt{fail} = \mathtt{fail}$, then $(\lambda_\emptyset\ \mathtt{a}\ \mathtt{b}\ \mathtt{b}\ .\widehat{y})(\mathtt{a}\ \Omega\ \mathtt{c})$, where Ω is a non-terminating term, would never fail as expected.

A (β) **redex** is a term of the form $(\lambda_\theta\ p.s)u$ s.t. $\{p/_\theta\ u\}$ is decided. A redex $(\lambda_\theta\ p.s)u$ s.t. $\{p/_\theta\ u\} = \mathtt{fail}$ is called a **matching failure**. A position $a \in \mathtt{Pos}(t)$ is called a **redex occurrence** of t iff $t|_a$ is a redex; $\mathcal{RO}(t)$ denotes the **set of all redex occurrences** of a term t. A redex in t is **outermost** if it is not contained in any other redex; *i.e.* it is minimal with respect to the order $<$ on redex occurrences. A **reduction step from t to s via a** is a tuple $t \stackrel{a}{\to} s$, where $t|_a$ is a redex $(\lambda_\theta\ p.u)v$ and $s = t[\{p/_\theta\ v\}u]_a$; this reduction step denotes the *contraction* (or *evaluation*) of the redex occurrence $a \in \mathcal{RO}(t)$. We occasionally identify a reduction step $t \stackrel{a}{\to} s$ with the redex occurrence a if no confusion arises.

Given a sequence of positions $\delta = a_1; \ldots; a_n; \ldots$ (possibly empty or infinite) and $t_0 \in \boldsymbol{T}$, a **reduction sequence from t_0 via δ**, written $t_0 \stackrel{\delta}{\twoheadrightarrow}$, is a sequence of the form $t_0 \stackrel{a_1}{\to} t_1 \ldots t_{n-1} \stackrel{a_n}{\to} t_n \ldots$. We write `nil` for the empty reduction sequence. We occasionally identify $t \stackrel{\delta}{\twoheadrightarrow} s$ with δ if no confusion arises. The term t **reduces to s in many steps**, written $t \twoheadrightarrow s$, iff there is a reduction sequence $t \stackrel{\delta}{\twoheadrightarrow} s$. Notice that $\twoheadrightarrow$ is the reflexive and transitive closure of $\to$. Given $t_0 \stackrel{\delta}{\twoheadrightarrow}$, where $\delta = a_1; \ldots; a_n; \ldots$, we write $\delta[i..k]$ $(1 \leq i \leq k \leq n)$ to identify the finite (sub)reduction sequence $t_{i-1} \stackrel{a_i}{\to} t_i \ldots t_{k-1} \stackrel{a_k}{\to} t_k$ and $\delta[i]$ $(1 \leq i \leq n)$ to identify the reduction step $t_{i-1} \stackrel{a_i}{\to} t_i$.

A term s is in **normal form**, written $s \in \boldsymbol{NF}$, iff there is no t s.t. $s \to t$. A term s is **normalising** iff there is a normal form t s.t. $s \twoheadrightarrow t$.

We refer the interested reader to [11] where different `PPC` examples are introduced, particularly to illustrate path and pattern polymorphism.

[3] A pattern p is linear w.r.t. θ if for every x in θ, the matchable $\widehat{x}$ appears at most once in p.
[4] Note that the notation for (compound) matching we have just given differs from [10] and [11]: the pattern and argument appear in reversed order there.

3 Multisteps and multireductions

In the light of the discussion in the Introduction on the inexistence of needed redexes, the reduction strategy for PPC we shall propose in Sec. 4 will select a *set* of redexes to contract at each step. Contraction of a set of redexes is understood as *simultaneous* contraction of all its members. Since, in principle, the order in which these members are contracted could affect the target term of the step, it becomes necessary to lay out precise definitions of what it means to perform simultaneous contraction of a set of redexes. It should be mentioned that these definitions are rather straightforward in first-order rewriting since the aforementioned set of redexes may be assumed to contain *pairwise disjoint* redexes, without any loss of generality. This owes to the fact that (first-order) residuals of such sets are again pairwise disjoint. In the higher-order case, however, this no longer holds, as can be seen by means of the following example. Consider the reduction step

$$t = (\lambda_{\{x\}}\ \widehat{x}.(\lambda_{\{y\}}\ \widehat{y}.y\ x)s)r \rightarrow (\lambda_{\{y\}}\ \widehat{y}.y\ r)s = u$$

where r is a redex and s an arbitrary term; the redexes $(\lambda_{\{y\}}\ \widehat{y}.y\ x)s$ and r are disjoint in t, while $(\lambda_{\{y\}}\ \widehat{y}.y\ r)s$ and r, their respective residuals in u (according to the formal definition given below), are not. This significantly complicates any effort of adapting extant results on normalisation of first-order systems to the higher-order setting.

Given a term t, $b \in \texttt{Pos}(t)$ and $a \in \mathcal{RO}(t)$, the **descendants of** b **after** a **in** t, written $b/^t a$ or simply b/a if the term is clear from the context, is the set of *positions* defined as follows:

$$\begin{array}{rl} \emptyset & \text{if } \ a = b. \\ \{b\} & \text{if } \ a \not\leq b. \\ \{an\} & \text{if } \ b = a12n, t|_a = (\lambda_\theta\ p.s)u, \text{ and } \{p/_\theta\ u\} \text{ is a substitution} \\ \{akn\ .\ s|_k = x\} & \text{if } \ b = a2mn, t|_a = (\lambda_\theta\ p.s)u, \{p/_\theta\ u\} \text{ is a substitution}, p|_m = \widehat{x} \text{ and } x \in \theta \\ \emptyset & \text{otherwise} \end{array}$$

If b is the position of a redex in t, then each position in b/a denotes a **residual** of b after performing a. This notion is extended to sets $\mathcal{B} \subseteq \mathcal{RO}(t)$ as follows: the **residuals of** $\mathcal{B}$ **after** a **in** t are $\mathcal{B}/a := \bigcup_{b \in \mathcal{B}} b/a$. In particular $\emptyset/a = \emptyset$. Given $t \overset{\delta}{\twoheadrightarrow} u$ and $\mathcal{B} \subseteq \mathcal{RO}(t)$, the **residuals of** $\mathcal{B}$ **after the sequence** δ, are: if $\delta = \texttt{nil}$, then $\mathcal{B}/\delta := \mathcal{B}$; otherwise $\mathcal{B}/\delta := (\mathcal{B}/\delta[1])/\delta[2\ldots n]$ where $\delta = \delta[1..n]$.

Let $\mathcal{A} \subseteq \mathcal{RO}(t)$. The reduction sequence $t \overset{\delta}{\twoheadrightarrow}$ is a **development** of $\mathcal{A}$ iff $\delta[i] \in \mathcal{A}/\delta[1..i-1]$ for all i. A development δ of $\mathcal{A}$ is said to be **complete** iff δ is finite and $\mathcal{A}/\delta = \emptyset$.

Using standard techniques we show that the order in which contraction of redexes in a given set is performed does not introduce non-termination nor does it affect the target term.

▶ **Proposition 1** (Strong Finite Developments). *Let t be a term and $\mathcal{A} \subseteq \mathcal{RO}(t)$.*

(i) All developments of $\mathcal{A}$ from t are finite.

(ii) If $t \overset{\delta_i}{\twoheadrightarrow} u_i$ $(i = 1, 2)$ are two complete developments of $\mathcal{A}$, then $u_1 = u_2$ and $\forall a \in \mathcal{RO}(t)$, $a/\delta_1 = a/\delta_2$.

As a consequence, for every $\mathcal{A} \subseteq \mathcal{RO}(t)$ we can now formally define a **multistep** $t \overset{\mathcal{A}}{\multimap} u$ iff there is a complete development of $\mathcal{A}$ from t to u. Given a sequence of sets $\Delta = \mathcal{A}_1; \ldots; \mathcal{A}_n; \ldots$ (possibly empty or infinite) and $t_o \in \boldsymbol{T}$, a **multireduction sequence** Δ from t_0, written $t_0 \overset{\Delta}{\multimap\!\!\!\twoheadrightarrow}$, is a sequence of the form $t_0 \overset{\mathcal{A}_1}{\multimap} t_1 \ldots t_{n-1} \overset{\mathcal{A}_n}{\multimap} t_n \ldots$. We write nil for the empty multireduction sequence. We occasionally identify $t \overset{\Delta}{\multimap\!\!\!\twoheadrightarrow} s$ with Δ if no confusion arises. We use the notations $\Delta[i]$ and $\Delta[i..j]$ to denote $\mathcal{A}_i$ and the (sub)sequence $\mathcal{A}_i; \ldots; \mathcal{A}_j$ respectively. We use $\Gamma, \Delta, \Pi, \Psi, \ldots$ to denote multireduction sequences. Let $\mathcal{A}$, $\mathcal{B}$ be sets of positions. The **residual of $\mathcal{B}$ after $\mathcal{A}$**, written $\mathcal{B}/\mathcal{A}$, is defined as the multistep $\mathcal{B}/\delta$ where δ is *any* complete development of $\mathcal{A}$ (this is well-defined by Prop. 1).

We extend the concept of residual to multireductions as well. The **residual of $\mathcal{B}$ after Δ** is defined as follows: if $\Delta = \texttt{nil}$, then $\mathcal{B}/\Delta := \mathcal{B}$; otherwise $\mathcal{B}/\Delta := (\mathcal{B}/\Delta[1])/\Delta[2..n]$ where $\Delta = \Delta[1..n]$. Analogously, we define the **residual of Δ after $\mathcal{B}$** as follows: $\texttt{nil}/\mathcal{B} := \texttt{nil}$, $\Delta[1..n]/\mathcal{B} := (\Delta[1]/\mathcal{B})/(\Delta[2..n]/(\mathcal{B}/\Delta[1]))$. By applying several times Prop. 1 it can be proved that $\Delta; (\mathcal{B}/\Delta)$ and $\mathcal{B}; (\Delta/\mathcal{B})$ end in the same term and induce the same residual relation.

Prop. 1 also allows us to introduce the **depth** of $\mathcal{A}$, written $\nu(\mathcal{A})$, a notion we shall use in Sec. 5. It is defined as the length of the longest complete development of $\mathcal{A}$. Since `PPC` is finitely branching, this is well-defined by König's Lemma. We occasionally use the notation $\nu(\mathcal{A}, t)$ to make explicit the source of the multistep $\mathcal{A}$.

4 The reduction strategy

The rationale behind the reduction strategy for `PPC` is that rather than selecting the entire set of outermost redexes of a given term t, this set is *refined* in two complementary ways. Let us call **preredex** a term of the form $(\lambda_\theta\, p.t)u$, regardless of whether the match $\{p/_\theta\, u\}$ is decided or not. The first observation about the strategy is that it focuses on the leftmost-outermost (LO) preredex of t, entailing that when `PPC` is restricted to the λ-calculus it behaves exactly as the LO strategy for the λ-calculus. Second, if the match corresponding to the LO occurrence of a preredex is not decided, then the strategy selects only the (outermost) redexes in that subterm which should be contracted to get it "closer" to a decided match.

Suppose $(\lambda_\theta\, p.t)\, u$ is this LO preredex. If $\{p/_\theta\, u\}$ is decided, then the preredex (in fact a redex), is the only one selected by the strategy (it is LO in this case). If the match $\{p/_\theta\, u\}$ is not decided, then the strategy selects the outermost redexes whose contraction may *contribute* towards obtaining a decided match. More precisely, in the term $(\lambda_{\{x,y\}}\, \texttt{a}\, \widehat{x}\, (\texttt{c}\, \widehat{y}).y\, x)\, (\texttt{a}\, r_1\, r_2)$ the match $\{\texttt{a}\, \widehat{x}\, (\texttt{c}\, \widehat{y})/_{\{x,y\}}\, \texttt{a}\, r_1\, r_2\}$ is not decided and the role played by r_1 is different from that of r_2 in obtaining a decided match. Replacing r_1 by an arbitrary term t_1 will not yield a decided match, *i.e.* $\{\texttt{a}\, \widehat{x}\, (\texttt{c}\, \widehat{y})/_{\{x,y\}}\, \texttt{a}\, t_1\, r_2\}$ is not decided. However, replacing r_2 by $\texttt{c}\, s_2$ (resp. by $\texttt{d}\, s_2$) does: $\{\texttt{a}\, \widehat{x}\, (\texttt{c}\, \widehat{y})/_{\{x,y\}}\, \texttt{a}\, r_1\, (\texttt{c}\, s_2)\} = \{x \to r_1, y \to s_2\}$ (resp. $\{\texttt{a}\, \widehat{x}\, (\texttt{c}\, \widehat{y})/_{\{x,y\}}\, \texttt{a}\, r_1\, (\texttt{d}\, s_2)\} = \texttt{fail}$). Hence, contraction of r_2 can contribute towards obtaining a decided match, while contraction of r_1 does not.

The selection of redexes contributing towards a decided match is performed by a simultaneous structural analysis of both pattern and argument. Since `PPC` allows patterns to be reduced, the selected redexes can lie inside a pattern or an argument of a preredex; and not in the body of the abstraction. Take *e.g.* $(\lambda_{\{x,y\}}\, \texttt{a}\, (\texttt{b}\, \widehat{x})\, r_1.r_2)\, (\texttt{a}\, r_3\, (\texttt{d}\, r_4))$ where every r_i is a redex. The strategy selects r_1 and r_3. Moreover, notice that contraction of r_4 is delayed since r_1 is not in matchable form (if the contractum of r_1 were *e.g.* either $\texttt{d}\, \widehat{y}$ or $\texttt{a}$, then the match w.r.t. $\texttt{d}\, r_4$ would be decided without the need of reducing r_4).

The reduction strategy $\mathcal{S}$ is defined as a function from terms to *sets of positions* of redex occurrences which should be reduced as a multistep. It is defined simultaneously with an auxiliary function $\mathcal{SM}$ from sets of symbols and pairs of terms to *pairs of sets of positions* which models the selection of contributing redexes towards a decided match.

$$
\begin{array}{rll}
\mathcal{S}(x) := & \emptyset & \\
\mathcal{S}(\widehat{x}) := & \emptyset & \\
\mathcal{S}(\lambda_\theta\, p.t) := & 1\mathcal{S}(p) & \text{if } p \notin \boldsymbol{NF} \\
\mathcal{S}(\lambda_\theta\, p.t) := & 2\mathcal{S}(t) & \text{if } p \in \boldsymbol{NF} \\
\mathcal{S}((\lambda_\theta\, p.t)u) := & \{\epsilon\} & \text{if } \{p/_\theta\ u\} \text{ decided} \\
\mathcal{S}((\lambda_\theta\, p.t)u) := & 11G \cup 2D & \text{if } \{p/_\theta\ u\} = \texttt{wait}, \mathcal{SM}_\theta(p,u) = \langle G, D\rangle \neq \langle\emptyset,\emptyset\rangle, \\
\mathcal{S}((\lambda_\theta\, p.t)u) := & 11\mathcal{S}(p) & \text{if } \{p/_\theta\ u\} = \texttt{wait}, \mathcal{SM}_\theta(p,u) = \langle\emptyset,\emptyset\rangle, p \notin \boldsymbol{NF} \\
\mathcal{S}((\lambda_\theta\, p.t)u) := & 12\mathcal{S}(t) & \text{if } \{p/_\theta\ u\} = \texttt{wait}, \mathcal{SM}_\theta(p,u) = \langle\emptyset,\emptyset\rangle, p \in \boldsymbol{NF}, t \notin \boldsymbol{NF} \\
\mathcal{S}((\lambda_\theta\, p.t)u) := & 2\mathcal{S}(u) & \text{if } \{p/_\theta\ u\} = \texttt{wait}, \mathcal{SM}_\theta(p,u) = \langle\emptyset,\emptyset\rangle, p \in \boldsymbol{NF}, t \in \boldsymbol{NF} \\
\mathcal{S}(tu) := & 1\mathcal{S}(t) & \text{if } t \text{ is not an abstraction and } t \notin \boldsymbol{NF} \\
\mathcal{S}(tu) := & 2\mathcal{S}(u) & \text{if } t \text{ is not an abstraction and } t \in \boldsymbol{NF}
\end{array}
$$

$$
\begin{array}{rll}
\mathcal{SM}_\theta(\widehat{x}, t) := & \langle\emptyset,\emptyset\rangle & \text{if } x \in \theta \\
\mathcal{SM}_\theta(\widehat{x}, \widehat{x}) := & \langle\emptyset,\emptyset\rangle & \text{if } x \notin \theta \\
\mathcal{SM}_\theta(p_1p_2, t_1t_2) := & \langle 1G_1 \cup 2G_2, 1D_1 \cup 2D_2\rangle & \text{if } t_1t_2, p_1p_2 \in \boldsymbol{MF}, \mathcal{SM}_\theta(p_i, t_i) = \langle G_i, D_i\rangle \\
\mathcal{SM}_\theta(p, t) := & \langle \mathcal{S}(p), \emptyset\rangle & \text{if } p \notin \boldsymbol{MF} \\
\mathcal{SM}_\theta(p, t) := & \langle \emptyset, \mathcal{S}(t)\rangle & \text{if } p \in \boldsymbol{MF}\ \&\ t \notin \boldsymbol{MF}\ \&\ \neg(p = \widehat{x}\ \&\ x \in \theta)
\end{array}
$$

The auxiliary function $\mathcal{SM}$ formalises the simultaneous structural analysis of the argument and pattern of a preredex. Its outcome is a pair of sets of positions, corresponding to redexes inside the pattern and argument respectively, which could contribute to turning a non decided match into a decided one. Notice the similarities between the first three clauses in the definition of $\mathcal{SM}$ and those of the definition of the matching operation (*cf.* Sec. 2).

If the LO preredex of a term is in fact a redex, then the strategy selects exactly that redex (fifth clause); if it is not a redex, and the function $\mathcal{SM}$ returns some redexes which could contribute towards a decided match, then the strategy selects them (sixth clause).

Otherwise, the preredex will never turn into a redex. Indeed, it can be proved that, given p and u such that $\{p/_\theta\ u\} = \texttt{wait}$, if there exist p' and u' such that $p \twoheadrightarrow p'$, $u \twoheadrightarrow u'$ and $\{p'/_\theta\ u'\}$ is decided, then $\mathcal{SM}_\theta(p,u) \neq \langle\emptyset,\emptyset\rangle$. In this case the strategy looks for the LO preredex inside the components of the term (seventh, eighth and ninth clauses).

The remaining clauses in the definition of $\mathcal{S}$ formalise the focus on the LO preredex for other forms of the term.

Returning to the example t_2 given in Sec. 1, namely $(\lambda_{\{y\}}\ \texttt{a b c}\ \widehat{y}.y)\ (\texttt{a}\ (\texttt{I c})\ (\texttt{I b})\ (\texttt{I a}))$, notice that the set $\{\texttt{I c}, \texttt{I b}\}$ is selected by the strategy $\mathcal{S}$, even if the contraction of just one redex of the set suffices to make the head match decided as explained before.

The reduction strategy $\mathcal{S}$ is complete. Formally:

▶ **Lemma 2.** *Let $t \notin \boldsymbol{NF}$. Then $\mathcal{S}(t) \neq \emptyset$, and $\mathcal{S}(t)$ only contains outermost redexes in t.*

Moreover, notice that $\mathcal{S}$ is not *outermost fair* [23]. Indeed, given $(\lambda\,\texttt{c}\,x.s)\,\Omega$, where Ω is a non-terminating term, $\mathcal{S}$ continuously contracts Ω, even when s contains a redex.

5 The reduction strategy $\mathcal{S}$ is normalising

This section proves that $\mathcal{S}$ is normalising for PPC. Our proof is mainly inspired from [21] which proves that selecting *necessary sets* (as introduced in Sec. 1) is a sufficient condition for a reduction strategy to be normalising in a first-order setting.

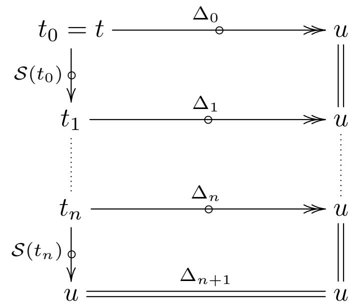


Figure 4 Proof idea.

The proof proceeds as follows: given any multireduction to normal form starting from a term t_0, say $t_0 \overset{\Delta_0}{\twoheadrightarrow} u \in \boldsymbol{NF}$, we construct another multireduction starting from t_0 to normal form having the form $t_0 \overset{\mathcal{S}(t_0)}{\rightarrow} t_1 \overset{\Delta_1}{\twoheadrightarrow} u$; where $\mathcal{S}(t_0)$ is the set of redexes of t_0 chosen by the strategy $\mathcal{S}$ and the multireduction Δ_1 is strictly smaller than the original one w.r.t. a convenient well-founded ordering. Well-foundedness of the ordering entails that repeated evaluation of the set of redexes selected by the strategy $\mathcal{S}$ yields the normal form u. This is depicted in Fig. 4 where Δ_{k+1} is strictly smaller than Δ_k for all k and Δ_{n+1} is a trivial multireduction. Thus, the original multireduction Δ_0 is first transformed into $\mathcal{S}(t_0); \Delta_1$, then successively into $\mathcal{S}(t_0); \ldots; \mathcal{S}(t_k); \Delta_{k+1}$; and finally into $\mathcal{S}(t_0); \ldots; \mathcal{S}(t_n)$.

Some difficulties arise when developing such a proof for higher-order calculi like `PPC`, particularly in two aspects: it is not trivial to show that the succesive multireductions which are built during the proof have the same target; nor is it straightforward to show that the sequence of multireductions $\Delta_0, \ldots, \Delta_k, \ldots$ is strictly decreasing.

To handle the first of these problems, we use a technique consisting of postponement of certain (non relevant) redexes with respect to other (relevant) ones (our notion of relevant is however different from that appearing in [22]). As for the second problem we define a measure inspired from [21, 22]. In order to prove that the sequence $\Delta_0, \ldots, \Delta_k, \ldots$ is strictly decreasing w.r.t. this measure, we need to resort to an additional property verified by the sets of redexes selected by $\mathcal{S}$, namely that they are *non-gripping*. In the following, we formalise the concept of necessary sets of redexes, we motivate and introduce the additional non-gripping property, and finally we give a brief outline of the proof.

5.1 Necessary sets of redexes

As explained in Sec. 1, there are some terms in `PPC` which do not have any *needed* redex; this was illustrated by the term $t_2 := (\lambda_{\{y\}}$ `a b c` $\widehat{y}.y)$ (`a (I c) (I b) (I a)`) whose (outermost) redexes are `I c`, `I b` and `I a`: it is possible to construct a reduction sequence from t_2 to normal form which ignores either of these redexes.

To formalise the meaning of needed redex, we first resort to the following notion of ignoring (or not) a redex along a (multi)reduction: given a term t and $\mathcal{B} \subseteq \mathcal{RO}(t)$, a multistep/multireduction from t *uses* $\mathcal{B}$ iff it contracts at least one redex or one residual of a redex in $\mathcal{B}$. Formally, let t be a term, $b \in \mathcal{RO}(t)$, $\mathcal{B} \subseteq \mathcal{RO}(t)$, a multistep $t \overset{\mathcal{A}}{\rightarrow} u$ and a multireduction $t \overset{\Delta}{\twoheadrightarrow} u$. Then, $\mathcal{A}$ **uses** b iff $b \in \mathcal{A}$; Δ **uses** b iff $\Delta[k] \cap (b/\Delta[1..k-1]) \neq \emptyset$ for at least one k; $\mathcal{A}$ (resp. Δ) **uses** $\mathcal{B}$ iff it uses at least one $b \in \mathcal{B}$. Now, given a term t, a redex $a \in \mathcal{RO}(t)$ is **needed** for t iff every reduction from t to normal form uses a.

Notice that the above term t_2 has no needed redex. Now consider the *set* of redexes $\{$`I c`$,$ `I b`$\}$ in the same term t_2. All reductions from t_2 to normal form must use at least one of these redexes *i.e. the set of redexes* $\{$`I c`$,$ `I b`$\}$ as a whole cannot be ignored to obtain the normal form of t_2. We formalise this notion as follows: given a term t, a set $\mathcal{A} \subseteq \mathcal{RO}(t)$ is a **necessary set** for t, iff every reduction from t to normal form uses $\mathcal{A}$.

The notion of necessary set generalises that of needed redex (notice that any singleton whose only element is a needed redex is a necessary set). There is, however, an important difference: while not all terms admit a needed redex, any term admits at least one necessary set, *i.e.* the set of *all* its redexes.

The reduction strategy $\mathcal{S}$ defined in Sec. 4 selects *necessary sets* of redexes; this property turns out to be crucial to prove that $\mathcal{S}$ is normalising. Formally:

▶ **Proposition 3.** *Let t be a term such that $t \notin \mathbf{NF}$. Then $\mathcal{S}(t)$ is a necessary set for t.*

5.2 Gripping

To motivate the notion of *gripping* [16], let us consider the following example, suggested by V. van Oostrom:

$$t_5 := \left((\lambda_{\{x\}}\widehat{x}.\,D\;x^{b})\;(\mathtt{I}\;y)^{a_2}\right)^{a_1} \xrightarrow{\mathcal{B}}\!\!\!\circ\!\!\rightarrow \left((\lambda_{\{x\}}\widehat{x}.x\;x)\;(\mathtt{I}\;y)^{a_2}\right)^{a_1} = u_5$$

where $\mathcal{A} := \{a_1, a_2\}$, $\mathcal{B} := \{b\}$ and $D := \lambda_{\{x\}}\;\widehat{x}.x\;x$. It is easy to verify that $\mathcal{A}/\mathcal{B} = \mathcal{A}$. Nevertheless, the *depth* (*cf.* Sec. 3) of $\mathcal{A}$ does change: the set $\mathcal{A}$ admits a development from u_5 requiring two contractions of $\mathtt{I}\;y$, while this is not the case for t_5; yielding $\nu(\mathcal{A}, t_5) = 2 < 3 = \nu(\mathcal{A}, u_5)$.

The notion of gripping turns out to be appropiate to explain this example. Informally, a redex b *grips* another redex a iff $a < b$ and there are occurrences of variables inside b which are bound by the abstraction of the redex a. In such a case, b-reduction may duplicate or erase those variable occurrences, thus affecting the depth of multisteps including a. In the term t_5 above, the redex b grips the redex a_1 because x, ocurring inside b, is bound by the abstraction of a_1; duplication of x explains the depth increase of $\mathcal{A}$ from u_5.

The following definition formalises the notion of gripping for PPC: given $a, b \in \mathcal{RO}(t)$, say $t|_a = (\lambda_\theta p.s)u$, we say that b **grips** a iff $\{p/_\theta\; u\} \neq \mathtt{fail}$, $a12 \leq b$ and $\mathtt{fv}(t|_b) \cap \theta \neq \emptyset$. Given $\mathcal{A}, \mathcal{B} \subseteq \mathcal{RO}(t)$, we say that $\mathcal{B}$ **grips** $\mathcal{A}$ iff there exist $b \in \mathcal{B}$ and $a \in \mathcal{A}$ such that b grips a.

In addition, we define a set $\mathcal{B} \subseteq \mathcal{RO}(t)$ to be **non-gripping** in t (or just **non-gripping** if t is clear from the context) iff for any multireduction Ψ such that $t \xrightarrow{\Psi}\!\!\!\circ\!\!\twoheadrightarrow u$, $\mathcal{B}/\Psi$ does not grip $\mathcal{RO}(u)$. Notice that when $\mathcal{B}$ is non-gripping all its residuals are.

The reduction strategy $\mathcal{S}$ defined in Sec. 4 selects *non-gripping* sets of redexes, formally

▶ **Proposition 4.** *Let t be a term. Then $\mathcal{S}(t)$ is non-gripping in t.*

Proof. (sketch) One first proves that $t \xrightarrow{\Psi}\!\!\!\circ\!\!\twoheadrightarrow u$, $a \in \mathcal{RO}(u)$, $b \in \mathcal{S}(t)/\Psi$ and $a < b$ imply a is a matching failure. This can be done by induction on the size of t, by considering the different cases in the (mutually recursive) definitions of $\mathcal{S}$ and $\mathcal{SM}$.

Now, let $t \xrightarrow{\Psi}\!\!\!\circ\!\!\twoheadrightarrow u$. To prove that $\mathcal{S}(t)$ is non-gripping in t we need to show that $\mathcal{S}(t)/\Psi$ does not grip $\mathcal{RO}(u)$. Suppose $b \in \mathcal{S}(t)/\Psi$, $a \in \mathcal{RO}(u)$ and b grips a. Then in particular $a < b$ and a is not a matching failure. But the previous observation entails that a is a matching failure which leads to a contradiction. ◀

This property allows to resort to gripping in order to guarantee that the depth of sets of redexes is stable under reduction in the cases in which this stability is needed in the normalisation proof. Another approach, distinguishing between *essential* and *inessential* sets of redexes, is taken in [22].

5.3 The measure

A measure on multireductions is defined by using the notion of *depth* for multisteps, denoted by ν and defined at the end of Sec. 3. Given $\Delta = \Delta[1..n]$ and $t_{i-1} \xrightarrow{\Delta[i]} t_i$ for all i, we define $\chi(\Delta, t_0)$ as the n-tuple $\langle \nu(\Delta[n], t_{n-1}), \ldots, \nu(\Delta[1], t_0) \rangle$; the lexicographic order is used to compare (measures of) multireductions. Notice that this (well-founded) ordering allows only to compare multireductions having the same length; the minimal elements being all the n-uples of the form $\langle 0, \ldots, 0 \rangle$. Another observation is that whenever $\chi(\Delta, t) < \chi(\Gamma, s)$ then for all multireductions $t' \xrightarrow{\Pi} t$, $s' \xrightarrow{\Psi} s$ having the same length $\chi(\Pi; \Delta, t') < \chi(\Psi; \Gamma, s')$ holds.

As remarked in [22], the measure used in [21], based on sizes of multisteps rather than depths, is not well-suited for a higher-order setting.

Returning to the key of the normalisation proof (*cf.* beginning of Sec. 5), the definition of Δ_{k+1} based on Δ_k and $\mathcal{S}(t_k)$ must verify $\chi(\Delta_{k+1}, t_{k+1}) < \chi(\Delta_k, t_k)$.

In order to further describe how Δ_{k+1} will be defined, and particularly how the notion of gripping will be applied to guarantee that the measure actually decreases, some additional notions are needed. Given a term t and $\mathcal{B} \subseteq \mathcal{RO}(t)$, a multistep/multireduction from t is *free* from $\mathcal{B}$ iff it does not contract any redex equal to or below a redex (or residual of a redex) in $\mathcal{B}$ (so it only contracts redexes lying above or disjoint with $\mathcal{B}$ and its residuals); a multistep from t is *dominated* by $\mathcal{B}$ iff all their redexes lie below some redex in $\mathcal{B}$. Formally, let t be a term, $a, b \in \mathcal{RO}(t)$, $\mathcal{A}, \mathcal{B} \subseteq \mathcal{RO}(t)$, and a multireduction $t \xrightarrow{\Delta} u$. Then,

- $\mathcal{A}$ is **free** from $\mathcal{B}$ iff $\mathcal{A} \cap \mathcal{B} = \emptyset$ and there does not exist $a \in \mathcal{A}$ and $b \in \mathcal{B}$ such that $b < a$.
- Δ is **free** from $\mathcal{B}$ iff $\Delta[k]$ is free from $\mathcal{B}/\Delta[1..k-1]$ for all k.
- a is **dominated** by $\mathcal{B}$ iff $a \notin \mathcal{B}$ and $\exists b \in \mathcal{B}$ s.t. $b < a$.
- $\mathcal{A}$ is **dominated** by $\mathcal{B}$ iff $\forall a \in \mathcal{A}$, a is dominated by $\mathcal{B}$.

For example, consider the following term

$$r_1 \ r_2 \ (\mathtt{I}\ ((\lambda_{\{x\}}\ \mathtt{a}\ \widehat{x}.r_3)\ (\mathtt{a}\ r_4))^{r_6})^{r_5})$$

where every r_i is a redex, and $\mathcal{B} = \{r_1, r_6\}$. Then the set $\{r_2, r_5\}$ is free from $\mathcal{B}$, $\{r_3, r_4\}$ is dominated by $\mathcal{B}$, and $\{r_2, r_3\}$ is neither free from nor dominated by $\mathcal{B}$.

Notice that $\mathcal{A}$ free from $\mathcal{B}$ and $\mathcal{C}$ dominated by $\mathcal{B}$ imply $\mathcal{A}$ free from $\mathcal{C}$.

Returning to the example in Sec. 5.2 notice that $\mathcal{A}$ is free from $\mathcal{B}$ and $\mathcal{A}/\mathcal{B} = \mathcal{A}$, however, as remarked before, $\nu(\mathcal{A}, t_5) < \nu(\mathcal{A}, u_5)$. A free set of redexes is always preserved by reduction; moreover, gripping explains all the cases in which the depth changes. Formally,

▶ **Lemma 5.** *Let $\mathcal{A}, \mathcal{B} \subseteq \mathcal{RO}(t)$ s.t. $\mathcal{A}$ is free from $\mathcal{B}$. Then $\mathcal{A}/\mathcal{B} = \mathcal{A}$.*

▶ **Lemma 6.** *Let $\mathcal{A}, \mathcal{B} \subseteq \mathcal{RO}(t)$ s.t. $\mathcal{A}$ is free from $\mathcal{B}$ and $t \xrightarrow{\mathcal{B}} s$. If $\mathcal{B}$ does not grip $\mathcal{A}$, then $\nu(\mathcal{A}, t) = \nu(\mathcal{A}, s)$.*

5.4 The normalisation proof

In this seciton we give a proof of the main result of the paper, namely that the strategy $\mathcal{S}$ is normalising. The proof is based on the ideas described at the beginning of Sec. 5. The main auxiliary results used in the proof are also included. They formalise the construction of Δ_{k+1} (*cf.* Fig. 4), in their statements $\mathcal{B}$ can be considered to be (some residual of) $\mathcal{S}(t_k)$ and $\mathcal{A}, \mathcal{C}, \Delta$ to be Δ_k or parts of it.

▶ **Lemma 7.** *Let $\mathcal{B}, \mathcal{C} \subseteq \mathcal{RO}(t)$ s.t. $t \xrightarrow{\mathcal{C}} u$, and $\mathcal{A} \subseteq \mathcal{RO}(u)$ s.t. $\mathcal{C}$ is dominated by $\mathcal{B}$, $\mathcal{A}$ is free from $\mathcal{B}/\mathcal{C}$, and $\mathcal{B}$ is non-gripping. Then $\mathcal{A} \subseteq \mathcal{RO}(t)$, $\mathcal{A}$ is free from $\mathcal{B}$, and $\nu(\mathcal{A}, t) = \nu(\mathcal{A}, u)$.*

Proof. (sketch) To obtain $\mathcal{A} \subseteq \mathcal{RO}(t)$ and $\mathcal{A}$ free from $\mathcal{B}$, we reason by induction on $\nu(\mathcal{C}, t)$; so let $c \in \mathcal{C}$ and $\delta = c; \delta'$ be a complete development of $\mathcal{C}$ (so δ' is a complete development of $\mathcal{C}/c$), *i.e.* $t \xrightarrow{c} s \xrightarrow{\mathcal{C}/c} u$. It is not difficult to prove that $\mathcal{C}$ dominated by $\mathcal{B}$ and $c \in \mathcal{C}$ imply $\mathcal{C}/c$ dominated by $\mathcal{B}/c$, allowing to use the *i.h.* on $\mathcal{C}/c$ to obtain $\mathcal{A} \subseteq \mathcal{RO}(s)$ and $\mathcal{A}$ free from $\mathcal{B}/c$. Now, for any $a \in \mathcal{A} \subseteq \mathcal{RO}(s)$, it can be proved (by contradiction) that $a \in \mathcal{RO}(t)$ and a is free from $\mathcal{B}$. Thus, $\mathcal{A}$ is free from $\mathcal{B}$. Noticing that $\mathcal{B}$ being non-gripping (so $\mathcal{B}$ does not grip $\mathcal{A}$), $\mathcal{A}$ free from $\mathcal{B}$ and $\mathcal{C}$ dominated by $\mathcal{B}$ imply $\mathcal{C}$ does not grip $\mathcal{A}$, depth stability follows from Lem. 6. ◀

▶ **Lemma 8.** *Let $t \xrightarrow{\mathcal{C}} s \xrightarrow{\Delta} u$ and $\mathcal{B} \subseteq \mathcal{RO}(t)$ s.t. $\mathcal{B}$ is non-gripping, $\mathcal{C}$ is dominated by $\mathcal{B}$, Δ is free from $\mathcal{B}/\mathcal{C}$, and $\mathcal{B}/(\mathcal{C}; \Delta) = \emptyset$. Then $t \xrightarrow{\Delta} u$, Δ is free from $\mathcal{B}$, $\mathcal{B}/\Delta = \emptyset$ and $\chi(\Delta, t) = \chi(\Delta, s)$.*

Proof. We proceed by induction on the size of Δ.

Assume $\Delta = \texttt{nil}$. Then $\mathcal{B}/(\mathcal{C}; \Delta) = \mathcal{B}/\mathcal{C} = \emptyset$. We first show $\mathcal{B} = \emptyset$. Indeed, suppose that $\mathcal{B} \neq \emptyset$, let b be a minimal element of $\mathcal{B}$ w.r.t. the prefix order. It is straightforward to verify that $\{b\}$ is free from $\mathcal{C}$ (since $\mathcal{C}$ is dominated by $\mathcal{B}$), then Lem. 5 yields $\{b\}/\mathcal{C} = \{b\}$, contradicting $\mathcal{B}/\mathcal{C} = \emptyset$. Then $\mathcal{B} = \emptyset$, therefore $\mathcal{C} = \emptyset$, again since $\mathcal{C}$ is dominated by $\mathcal{B}$, hence $t = s = u$ and the conclusions are straightforward.

If $\Delta \neq \texttt{nil}$, then consider $t \xrightarrow{\mathcal{C}} s \xrightarrow{\Delta[1]} w \xrightarrow{\Delta[2..n]} u$. Lem. 7 gives $\Delta[1] \subseteq \mathcal{RO}(t)$, $\Delta[1]$ is free from $\mathcal{B}$ and $\nu(\Delta[1], t) = \nu(\Delta[1], s)$; moreover, since $\Delta[1]$ is free from $\mathcal{B}$ and $\mathcal{C}$ is dominated by $\mathcal{B}$, then $\Delta[1]/\mathcal{C} = \Delta[1]$ (*cf.* Lem. 5), so $(\mathcal{C}; \Delta[1])$ and $(\Delta[1]; (\mathcal{C}/\Delta[1]))$ are two complete developments of $\Delta[1] \cup \mathcal{C}$. Hence, Prop. 1 implies that $t \xrightarrow{\Delta[1]} s' \xrightarrow{\mathcal{C}/\Delta[1]} w \xrightarrow{\Delta[2..n]} u$.

In order to apply the *i.h.* we need to verify the corresponding hypotheses. By a patient analysis on residuals and positions we obtain that $\mathcal{C}/\Delta[1]$ is dominated by $\mathcal{B}/\Delta[1]$. Moreover, $\mathcal{B}$ non-gripping implies $\mathcal{B}/\Delta[1]$ non-gripping, and the remaining conditions can be easily obtained by noticing that $\mathcal{B}/(\Delta[1]; (\mathcal{C}/\Delta[1])) = \mathcal{B}/(\mathcal{C}; \Delta[1])$.

Therefore the *i.h.* can be applied, obtaining $s' \xrightarrow{\Delta[2..n]} u$, $\Delta[2..n]$ is free from $\mathcal{B}/\Delta[1]$, $\mathcal{B}/(\Delta[1]; \Delta[2..n]) = \emptyset$, and $\chi(\Delta[2..n], s') = \chi(\Delta[2..n], w)$. We conclude by combining these results with those obtained in the first paragraph. ◀

▶ **Lemma 9.** *Let $t \xrightarrow{\Delta} u$ and $\mathcal{B} \in \mathcal{RO}(t)$ s.t. $\Delta = \Delta[1..n]$, $\mathcal{B}$ is non-gripping, Δ does not use $\mathcal{B}$ and $\mathcal{B}/\Delta = \emptyset$. Then there exists a multireduction $\Gamma = \Gamma[1..n]$ such that $t \xrightarrow{\Gamma} u$, Γ is free from $\mathcal{B}$, $\mathcal{B}/\Gamma = \emptyset$ and $\chi(\Gamma, t) \leq \chi(\Delta, t)$.*

Proof. We proceed by induction on n.

If $n = 0$ then $\Delta = \emptyset$, therefore it suffices to take $\Gamma = \emptyset$.

If $n > 0$ then we consider $t \xrightarrow{\Delta[1]} s \xrightarrow{\Delta[2..n]} u$. By observing that $\mathcal{B}/\Delta[1]$ is non-gripping we can use the *i.h.* on $t \xrightarrow{\Delta[2..n]} u$, thus obtaining a multireduction $\Gamma_1 = \Gamma_1[1..n-1]$ such that $s \xrightarrow{\Gamma_1} u$, Γ_1 is free from $\mathcal{B}/\Delta[1]$, $(\mathcal{B}/\Delta[1])/\Gamma_1 = \emptyset$ and $\chi(\Gamma_1, s) \leq \chi(\Delta[2..n], s)$.

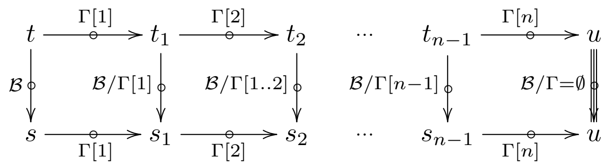
We now define $\Delta[1]^F := \{a \in \Delta[1] \text{ s.t. } \nexists b \in \mathcal{B} \,.\, b < a\}$ and $\Delta[1]^D := (\Delta[1] \setminus \Delta[1]^F)/\Delta[1]^F$, then $t \xrightarrow{\Delta[1]^F} t' \xrightarrow{\Delta[1]^D} s \xrightarrow{\Gamma_1} u$ for some term t'. It is easy to check that $\Delta[1]^F$ is free from $\mathcal{B}$ and $\Delta[1] \setminus \Delta[1]^F$ is dominated by $\mathcal{B}$ just by definition ($\Delta[1]$ does not use $\mathcal{B}$), and it can be proved that $\Delta[1]^D$ is dominated by $\mathcal{B}/\Delta[1]^F$. Moreover $\mathcal{B}/\Delta[1]^F$ is non-gripping. Finally, $\mathcal{B}/\Delta[1] = (\mathcal{B}/\Delta[1]^F)/\Delta[1]^D$ by Prop. 1; consequently Γ_1 is free from $(\mathcal{B}/\Delta[1]^F)/\Delta[1]^D$ and $(\mathcal{B}/\Delta[1]^F)/(\Delta[1]^D;\Gamma_1) = \emptyset$.

Therefore we can use Lem. 8 on $t' \xrightarrow{\Delta[1]^D} s \twoheadrightarrow^{\Gamma_1} u$, thus obtaining that $t' \xrightarrow{\Gamma_1}\!\!\!\!\twoheadrightarrow u$, Γ_1 is free from $\mathcal{B}/\Delta[1]^F$, $(\mathcal{B}/\Delta[1]^F)/\Gamma_1 = \emptyset$ and $\chi(\Gamma_1, t') = \chi(\Gamma_1, s) \leq \chi(\Delta[2..n], s)$.

We can conclude by taking $\Gamma := \Delta[1]^F;\Gamma_1$; notice that $\nu(\Delta[1]^F, t) \leq \nu(\Delta[1], t)$. ◀

▶ **Proposition 10.** *Let $t \xrightarrow{\Delta}\!\!\!\!\twoheadrightarrow u$ and $\mathcal{B} \subseteq \mathcal{RO}(t)$ s.t. $\mathcal{B}$ is non-gripping, Δ does not use $\mathcal{B}$, $\mathcal{B}/\Delta = \emptyset$ and $t \xrightarrow{\mathcal{B}} s$. Then $\exists\ \Gamma$ s.t. $s \xrightarrow{\Gamma}\!\!\!\!\twoheadrightarrow u$ and $\chi(\Gamma, s) \leq \chi(\Delta, t)$.*

Proof. Let us say $\Delta = \Delta[1..n]$. Lem. 9 yields the existence of some $\Gamma = \Gamma[1..n]$ such that $t \xrightarrow{\Gamma}\!\!\!\!\twoheadrightarrow u$, Γ is free from $\mathcal{B}$, $\mathcal{B}/\Gamma = \emptyset$ and $\chi(\Gamma, t) \leq \chi(\Delta, t)$. Let us define $t_0 := t$, $t_n := u$ and $t_{i-1} \xrightarrow{\Gamma[i]} t_i$ for all $i \leq n$. Notice that Γ being free from $\mathcal{B}$ implies that $\Gamma[i]/(\mathcal{B}/\Gamma[1..i-1]) = \Gamma[i]$ for all i, *cf.* Lem. 5. Therefore, we can build the following diagram



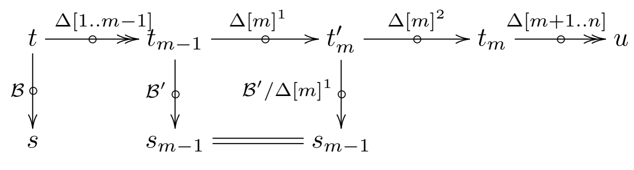
where for all i,

Lem. 6 yields $\nu(\Gamma[i], t_{i-1}) = \nu(\Gamma[i], s_{i-1})$ since $\mathcal{B}$ non-gripping implies that $\mathcal{B}/\Gamma[1..i-1]$ does not grip $\Gamma[i]$. We conclude by observing that $\chi(\Gamma, s) = \chi(\Gamma, t) \leq \chi(\Delta, t)$. ◀

▶ **Proposition 11.** *Let $t \xrightarrow{\Delta}\!\!\!\!\twoheadrightarrow u$ and $\mathcal{B} \subseteq \mathcal{RO}(t)$, s.t. $\mathcal{B}$ is non-gripping, Δ uses $\mathcal{B}$, $\mathcal{B}/\Delta = \emptyset$ and $t \xrightarrow{\mathcal{B}} s$. Then $\exists\ \Gamma$ s.t. $s \xrightarrow{\Gamma}\!\!\!\!\twoheadrightarrow u$ and $\chi(\Gamma, s) < \chi(\Delta, t)$.*

Proof. Let us say $\Delta = \Delta[1..n]$, $t_0 := t$, $t_n := u$ and $t_{i-1} \xrightarrow{\Delta[i]} t_i$ for all $i \leq n$. Since Δ uses $\mathcal{B}$, there exists some $\Delta[m]$ being the last step of Δ using (the corresponding residual of) $\mathcal{B}$. Formally, if $\mathcal{B}' := \mathcal{B}/\Delta[1..m-1]$, then $\Delta[m]^1 := \Delta[m] \cap \mathcal{B}' \neq \emptyset$ and $\Delta[m+1..n]$ does not use $\mathcal{B}/\Delta[1..m]$. Additionally, let $\Delta[m]^2 := (\Delta[m] \setminus \Delta[m]^1)/\Delta[m]^1$.

We can build the following diagram



since $\Delta[m]^1/\mathcal{B}' = \emptyset$.

Assume the existence of some $b \in \Delta[m]^2 \cap (\mathcal{B}'/\Delta[m]^1)$, this would imply that $b \in b'/\Delta[m]^1$ such that $b' \in (\Delta[m] \setminus \Delta[m]^1) \cap \mathcal{B}'$ since the ancestor of a redex is unique in `PPC`, contradicting the definition of $\Delta[m]^1$.

Therefore $\Delta' := \Delta[m]^2;\Delta[m+1..n]$ does not use $\mathcal{B}'/\Delta[m]^1$, hence Prop. 10 can be used to obtain some $\Pi = \Pi[1..n-m+1]$ such that $s_{m-1} \xrightarrow{\Pi}\!\!\!\!\twoheadrightarrow u$ and $\chi(\Pi, s_{m-1}) \leq \chi(\Delta', t'_m) < \chi(\Delta[m..n], t_{m-1})$.

We now define Γ as follows: $\Gamma[i] := \Delta[i]/(\mathcal{B}/\Delta[1..i-1])$ if $1 \leq i \leq m-1$, and $\Gamma[i] := \Pi[i-m+1]$ if $m \leq i \leq n$. We remark that Prop. 1 implies that $s \overset{\Gamma[1..m-1]}{\twoheadrightarrow} s_{m-1}$, thus Γ is well-defined. Moreover, the definition of the measure for multireductions by means of a *reversed* order implies that $\chi(\Pi, s_{m-1}) < \chi(\Delta[m..n], t_{m-1})$ is a sufficient condition to obtain $\chi(\Gamma, s) < \chi(\Delta, t)$. ◀

▶ **Theorem 12.** *The reduction strategy $\mathcal{S}$ is normalising.*

Proof. Let t_0 be a normalising term in PPC, then there exists some Δ_0 such that $t_0 \overset{\Delta_0}{\twoheadrightarrow} u$ and $u \in \boldsymbol{NF}$. We proceed by induction on $\chi(\Delta_0, t_0)$, using the well-founded ordering in Sec. 5.3.

If $t_0 \in \boldsymbol{NF}$ there is nothing to prove. Otherwise, Lem. 2 guarantees that $\mathcal{S}(t_0) \neq \emptyset$. Let $t_0 \overset{\mathcal{S}(t_0)}{\to} t_1$. Then Δ_0 uses $\mathcal{S}(t_0)$ and $\mathcal{S}(t_0)$ is non-gripping by Prop. 3 and Prop. 4 respectively; moreover, $u \in \boldsymbol{NF}$ implies $\mathcal{S}(t_0)/\Delta_0 = \emptyset$. Then Prop. 11 yields the existence of a multireduction Δ_1 s.t. $t_1 \overset{\Delta_1}{\twoheadrightarrow} u$ and $\chi(\Delta_1, t_1) < \chi(\Delta_0, t_0)$. We conclude by the *i.h.* ◀

As a final remark, notice that the construction of Δ_{k+1} from Δ_k and $\mathcal{S}(t_k)$ in the proof of Prop. 11 combines two different kinds of projections: one based on residuals (*cf.* Prop. 1), the other based in the notions of free and dominated sets of redexes.

6 Conclusions and further work

We study normalisation strategies for PPC, a dynamic pattern calculus equipped with *matching failure*. Its semantics induces a parallel-or-like, non-sequential behaviour which hinders the development of normalising strategies, particularly since it is not a FO system nor is it clear how it may be encoded in terms of established HO rewriting formalisms (eg. HRS [18], CRS [14]).

Building on ideas from [21] developed for FO systems, we propose a notion of *necessary set* of redexes for a HO language. Repeated contraction of necessary sets is shown to normalise a term *provided* that they are also *non-gripping* [16]. We introduce an inductively defined strategy that, given a term t, selects a necessary set of redexes for t which is also non-gripping, and moreover bounded by the set of outermost redexes. The strategy collapses to LO when the λ-calculus is encoded in PPC.

We think that our normalisation proof could be adapted to other (HO) calculi, and even to families of calculi defined in some general HO formalism, particularly since necessary and gripping sets are specified in a quite general way. Another research direction is to adopt a completely axiomatic approach, *e.g.* Abstract Rewriting Systems as defined in [16].

An encoding of PPC into some HO formalism, such as those mentioned above, could yield interesting insights on the possible transfer of the normalisation results of [23] and [22] from HORS to our framework.

Further avenues of research we intend to pursue include implementing an interpreter based on our strategy and devising even more refined strategies, in the sense of selecting smaller sets of redexes.

Acknowledgements To Vincent van Oostrom (for having pointed out a mistake in a previous version of this work), Femke van Raamsdonk and the anonymous referees (for helpful remarks). This work was partially funded by the Digiteo-IdF project MoDy and the French-Argentinian Laboratory in Computer Science **INFINIS**.

References

1 S. Antoy and A. Middeldorp. A sequential reduction strategy. *TCS*, 165(1):75–95, 1996.
2 F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge Univ. Press, 1998.
3 T. Balabonski. On the implementation of dynamic patterns. In *HOR*, EPTCS 49, pages 16–30, Edinburgh, UK, 2010.
4 T. Balabonski. Optimality for dynamic patterns: Extended abstract. In *PPDP*, pages 16–30. ACM, 2010.
5 H.P. Barendregt, R. Kennaway, J-W. Klop, and M. Ronan Sleep. Needed reduction and spine strategies for the lambda calculus. *I & C*, 75(3):191–231, 1987.
6 H. Cirstea and C. Kirchner. The rewriting calculus - Part I and Part II. *Logic Journal of the IGPL*, 9(3), 2001.
7 J. Glauert, R. Kennaway, and Z. Khasidashvili. Stable results and relative normalization. *JLC*, 10(3):323–348, 2000.
8 G. Huet and J-J Lévy. Computations in orthogonal rewriting systems - Parts I and II. In *Computational Logic, Essays in Honor of Alan Robinson*, pages 395–443. MIT Press, 1991.
9 B. Jay. *Pattern Calculus: Computing with Functions and Structures.* Springer, 2009.
10 B. Jay and D. Kesner. Pure pattern calculus. In *ESOP*, *LNCS* 3924, pages 100–114. Springer, 2006.
11 B. Jay and D. Kesner. First-class patterns. *JFP*, 19(2):191–225, 2009.
12 W. Kahl. Basic pattern matching calculi: A fresh view on matching failure. In *FLOPS*, *LNCS* 2998, pages 276–290. Springer, 2004.
13 D. Kesner, C. Lombardi, and A. Ríos. Standardisation for constructor based pattern calculi. In *HOR*, EPTCS 49, pages 58–72, Edinburgh, UK, 2010.
14 J-W. Klop. Combinatory Reduction Systems. *Mathematical Centre Tracts* 127. PhD thesis, University Amsterdam, 1980.
15 J-W. Klop, V. van Oostrom, and R. de Vrijer. Lambda calculus with patterns. *TCS*, 398(1-3):16–31, 2008.
16 P-A. Melliès. *Description abstraite des Systèmes de Réécriture.* PhD thesis, Université Paris VII, 1996.
17 P-A. Melliès. Axiomatic rewriting theory II: the $\lambda\sigma$-calculus enjoys finite normalisation cones. *JLC*, 10(3):461–487, 2000.
18 T. Nipkow. Higher-Order Critical Pairs. In *LICS*, pages 342–349, IEEE. 1991.
19 M. J. O'Donnell. *Computing in Systems Described by Equations*, *LNCS* 58. Springer, 1977.
20 S. L. Peyton-Jones. *The Implementation of Functional Programming Languages.* Prentice-Hall, Inc., 1987.
21 R. C. Sekar and I. V. Ramakrishnan. Programming in equational logic: Beyond strong sequentiality. *I & C*, 104(1):78–109, 1993.
22 V. van Oostrom. Normalisation in weakly orthogonal rewriting. In *RTA*, *LNCS* 1631, pages 60–74. Springer, 1999.
23 F. van Raamsdonk. Outermost-fair rewriting. In *TLCA*, *LNCS* 1210, pages 284–299. Springer, 1997.