Optimality for Dynamic Patterns

Thibaut Balabonski

Optimality for Dynamic Patterns

Full version

Thibaut Balabonski
Laboratoire PPS, CNRS and Université Paris Diderot
thibaut.balabonski@pps.jussieu.fr

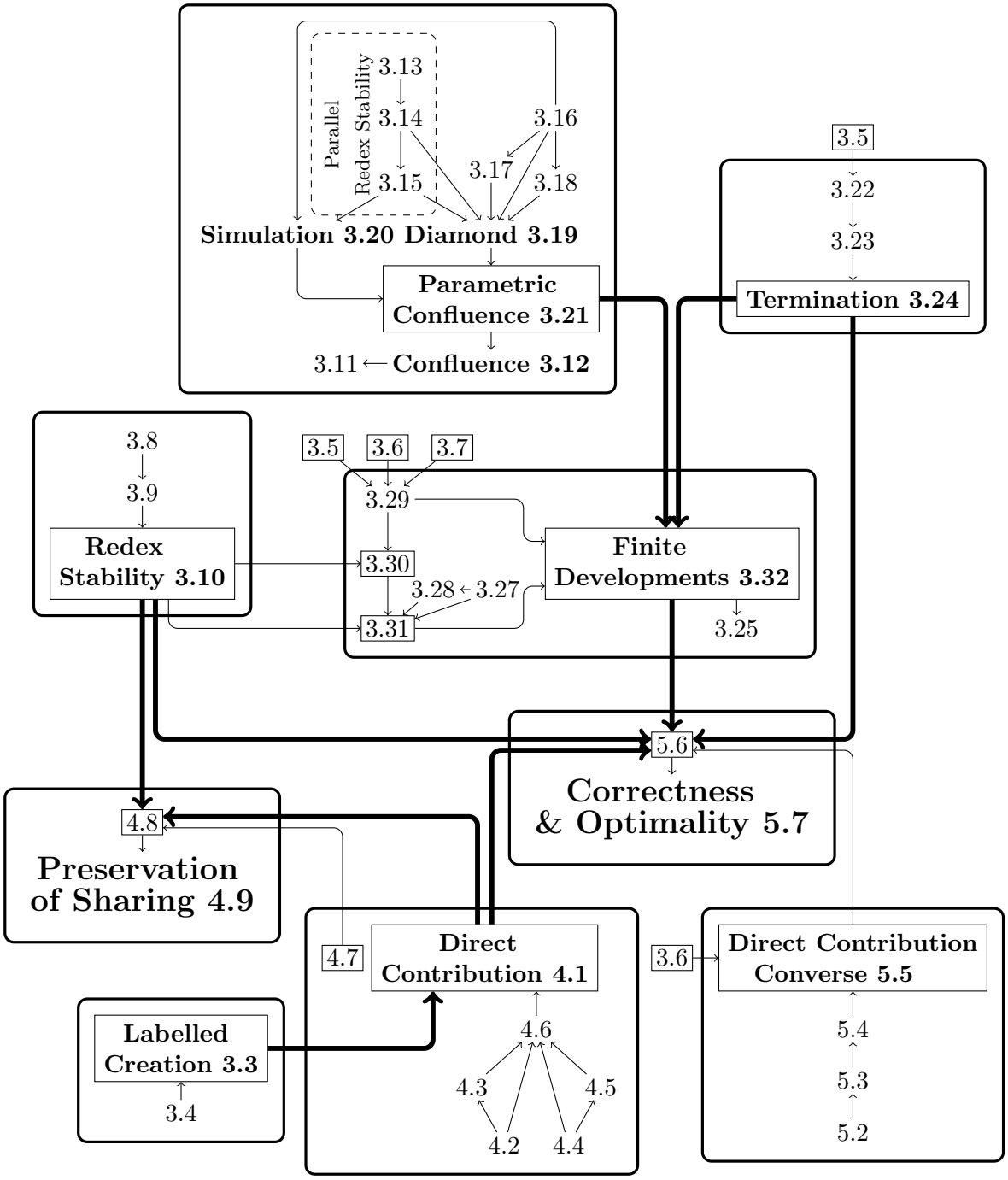
March 26, 2010

Abstract

Evaluation of a weak calculus featuring expressive pattern matching mechanisms is investigated by means of the construction of an efficient model of sharing. The sharing theory and its graph implementation are based on a labelling system derived from an analysis of causality relation between evaluation steps. The labelled calculus enjoys properties of confluence and finite developments, and is also used for proving correctness and optimality of a whole set of reduction strategies.

Contents

Proof Map



This picture shows all results mentioned in the paper. An arrow is drawn from $\texttt{Ref}_\texttt{a}$ to $\texttt{Ref}_\texttt{b}$ when $\texttt{Ref}_\texttt{a}$ is used to prove $\texttt{Ref}_\texttt{b}$. Each framed area is organized around a central result, and the framed references are those connected to external areas. Bold arrows emphasize dependencies between main results.

1 Introduction

The motivation of this work is to go toward an efficient implementation model for functional programming languages featuring expressive pattern matching facilities. As a first step, this paper studies sharing and optimality in a calculus with patterns.

Pattern matching can be simply understood as a basic mechanism used to define functions by cases on the structure of the argument; it is one of the main aspects revealing the interest of the functional paradigm for the working programmer. Unfortunately usual mechanisms suffer from some lack of genericity. Consider for instance a structure of binary tree: a tree is either a single data or a node with two subtrees, which could be written in ML-style as follows.

```
type 'a tree =
   | Data of 'a
   | Node of 'a tree * 'a tree
```

A function `upd` updating all data in binary trees can be easily written by recursion on the inductive structure of trees, using pattern matching.

```
let rec upd f t = match t with
  | Data a   -> Data (f a)
  | Node l r -> Node (upd l) (upd f r)
```

When the function `upd` is applied to arguments f and t, the latter is compared to the shape `Data a`, called a **pattern** (in the whole paper, bold font is used for definitions and technical vocabulary). In case of **success** (the pattern and the argument match, *i.e.* $t =$ `Data` u for some u), the **matching variable** `a` captures the substructure u of the argument t and then the updated data `Data` $(f(u))$ is returned. If the first comparison **fails** (pattern and argument don't match), then the **alternative case** `Node l r` is tested.

The code of `upd` names explicitly all the constructors which will be met during evaluation, that is, `Data` and `Node`. Thus, any minor variation on the data structure (trees of other or arbitrary arities, lists, heterogeneous data...) requires a new definition of the `upd` function.

A first solution for this problem is given by **path polymorphism** [Jay04, JK09], which allows a function to go recursively through any data structure and act on some base cases (like `Data a` here). Path polymorphism can be achieved by a universal –and very simple– classification of structures: each one is either atomic, or compound. Atoms, as constructors, are inert. Compounds, as (partially) applied constructors, lead to recursion. The code below shows how a path polymorphic `upd` could look like.

```
let rec upd f t = match t with
  | Data a -> Data (f a)            //Base
  | x y    -> (upd f x) (upd f y)   //Compound
  | z      -> z                     //Atom
```

The decomposition in compounds and atoms can be encoded in usual languages with the use of an explicit constructor for compounds. Note that this solution gives up typing. A more subtle approach is in Generic Haskell [HJ03] with reasoning by case on the type of the data structure.

Example 1.

Example of evaluation of the path polymorphic `upd` *(brackets are used as parentheses to separate function calls, and* `++` *is the successor function):*

```
                upd ++ (Node  (Data 1) (Data 2))
           [upd ++ (Node (Data 1))]  [upd ++ (Data 2)]
    [upd ++ Node] [upd ++ (Data 1)] [upd ++ (Data 2)]
            Node           (Data 2)          (Data 3)
```

While `Node` *alone is an atom, terms like* `Node (Data 1)` *and* `Node (Data 1) (Data 2)` *are compounds.*

The second solution to the previous problem is given by **pattern polymorphism** [JK09, Jay09], which allows to parametrize a pattern by an arbitrary term, thus allowing different instantiations of the same pattern. The simpler case is when the pattern parameter is just a constructor, the more general and interesting one is when the pattern parameter is a *function constructing a pattern*. Such a function needs to be evaluated before pattern matching is performed, thus patterns become **dynamic**. This is completely out of the scope of usual programming languages.

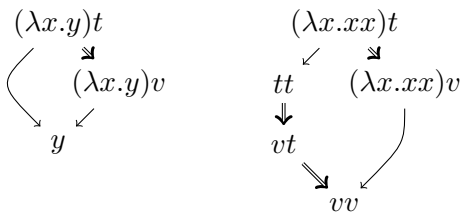
Dynamic patterns are really a key to expressive pattern matching paradigms, but they are incompatible with usual implementations. Indeed pattern matching definitions are normally compiled into series of low-level tests, organized in what is generically called a **matching automata**. Since a naive matching automata is likely to introduce a lot of redundant tests and other inefficiencies, sequences of patterns are analyzed in order to get optimized control structures which minimize the average cost of pattern matching and share redundant tests. See [FM01, Mar08] for examples of this. However, dynamic patterns are by nature undefined at compile-time and make this kind of static analysis impossible: new mechanisms have to be invented to recover some of the lost optimizations.

The aim of this work is to conciliate dynamic patterns and efficient evaluation by introducing run-time sharing mechanisms that minimize redundancy of matching operations.

Whereas both aspects are often separated, the study of sharing provided by this paper includes pattern matching in the kernel of functional programming languages (as can be found in higher-order rewriting). The framework used to model higher-order functions, data structures and pattern matching with dynamic patterns is the *Pure Pattern Calculus* (*PPC*) of Jay and Kesner [JK06, JK08, JK09, Jay09]. The formalism encompasses λ-calculus and usual pattern matching, allowing to apply any relevant result obtained here to the standard cases. Furthermore, this framework is even richer since it realizes both path and pattern polymorphisms, which is why it is preferred here to other pattern matching calculi such as the λ-calculus with patterns [Jon87, KvOdV08] or the rewriting calculus [Cir00].

Taking advantage of the unified framework of *PPC*, the discussion on efficiency led here concerns pattern matching as well as function calls. The paper thus proceeds by extending to the more general *PPC* some concepts of λ-calculus that are presented below.

A **reduction strategy** is the choice of an evaluation order for programs. Suppose the term t evaluates to the value v. The figure below illustrates how the choice of an evaluation order can have an impact on the number of evaluation steps, and then on the evaluation cost. In both pictures the left path is *call-by-name* while the right path is *call-by-value*.

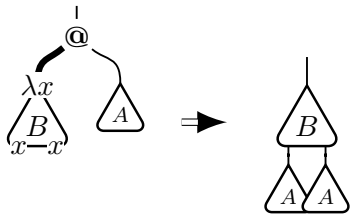


In the left picture, the difference comes from the presence of some useless fragment in the program, whereas in the right picture there is a possibility of duplication of a non-evaluated program. An efficient reduction strategy has to cope with these two pitfalls.

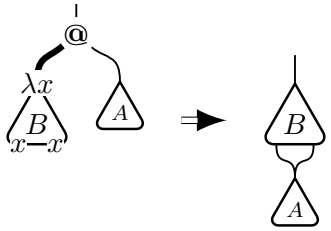
Whereas useless evaluation can be avoided (by so-called needed reduction), it is known that an evaluation order alone is not enough to prevent duplication [Lé80]. Thus an efficient implementation comes from the combination of a reduction strategy with a mechanism of **sharing** of duplicated terms. The idea is to make sure that some parts of a program which are *logically* duplicated (in the term representation of the program) remain *physically* single pieces (in the memory of the evaluator).

The use of sharing leads from classical term representations of programs (say, as higher-order rewriting systems [Ter03]) to graph representations [Jon87], where duplication of a whole subterm may be replaced by the copy of a single pointer, as showed below.

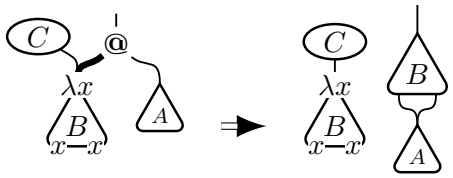
Pictures will take the form of syntactic trees or graphs, top-down oriented. Application is denoted by the binary node @, and redexes are marked with bold lines (**red***ucible* **ex***pression*, or **redex**, designs a place where an evaluation step can take place). For instance in the following picture, an abstraction $\lambda x.B$ is applied to an argument A. The function body B contains two occurences of the formal parameter x, and the argument A is thus logically duplicated, representing **call-by-name**:



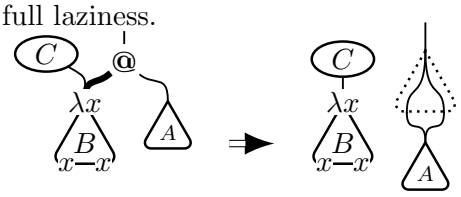
So-called **lazy** evaluation (next figure) prevents this duplication: A stays physically unique, with two pointers to its location.



The reader should keep in mind that the term laziness is to be taken in literal sense: if there is something to do (as a duplication), just wait for some better reason, which often appears quite fast. In particular a shared function has to be copied prior to any instantiation, as shown in the picture below.

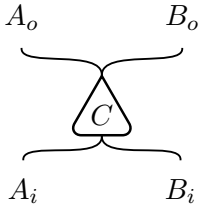


A further step in the hierarchy of lazinesses is called **fully lazy** evaluation, and performs only a partial duplication of the body of the function, namely the parts that depends on the formal parameter [Wad71]. In the next picture, the dotted zone represents the parts whose copy is (at least momentarily) saved by full laziness.



The graphs generated by this kind of reduction are directed acyclic graphs (DAGs) in which there is at most one arrow out of each node. An important consequence is that in this model, only complete subterms are shared, and the graphs are precise representations of what is present in the memory of the evaluator. Each node is a memory location, each edge is a pointer, and nothing more is needed for evaluation. It can also be noticed that any such graph can be easily *unfolded* (or **read-back**) into the term it represents.

Further levels of laziness [HG91, Lé80] lose these interesting properties. Indeed they need to share subcontexts (or open terms) instead of complete subterms [Sin08, AG98]: something which is shared at some point can be unshared later.



The problem here is to match correctly the inputs with the outputs. For correct evaluation the right associations have to be remembered, which needs to introduce some additional control structures. The possibly huge bookkeeping work induced by these additional structures [AG98] prevents optimal β-reduction from being the unquestionable optimal choice of implementation.

As a consequence, any really efficient model of sharing has to strike a balance between the effective amount of sharing and the additional implementation cost. The choice here is full laziness, as it is the higher known level of sharing that shares only complete subterms and then avoids the explosion of hidden costs that characterizes optimality. Moreover, this presumed efficiency of full laziness is effective: see for instance the implementation for λ-calculus proposed in [SW04].

All the reasoning on graph representations appearing in this paper is done through enriched terms representing graphs. To achieve this, terms are decorated with labels representing memory locations and pointers (see Section 4). Since memory locations determine what is shared or copied, the interesting point is the way in which the labels are designed (Section 3).

As in Lévy's optimality theory [Lé78], labels are meant to caracterize *equivalent* subterms (originally equivalent redexes) that should be shared (in other words *that should not have been separated*). Informally, being equivalent means to have a common origin, followed by equivalent evolutions (the whole being called **history**).

The kernel of this work is a representation of the history of terms by labels derived from an analysis of causality relations between evaluation events.

In the simple functional case a redex appears wherever an abstraction is applied to an argument, and the only relevant history in this case can be recorded locally as in [Lé78, BLM07]. However an argument that has to be matched against some pattern is required to be in a suitable form. Consider for instance the following program.

```
let id x = x
and rec count = fun
  | Data a    ->  1
  | Node l r  ->  (count l) + (count r)
in
  count (id (Data 0))
```

Function `count` is called with argument `id (Data 0)`, which has to be evaluated to `Data`

0 before the application can be resolved. Thereby a redex also depends on the history of its subterms, at an arbitrary depth.

These non-local contributions associated with a generic matching algorithm represent a new challenge and a major source of difficulty for this work, especially since pattern matching is allowed to fail. Moreover, in *PPC* this challenge concerns not only the arguments but also the (dynamic) patterns! Difficulty of this point is detailed in Subsection 3.1 (last paragraph) once all needed material is available, whereas novelty is argumented in Section 6 (related work).

Main originality of the paper is the advanced treatment of history and contributions, which is still unexplored for the kind of pattern matching presented here, as far as the author is aware.

As done in current implementations for functional languages, weak evaluation is considered here. More precisely, *PPC* will be restricted by constraining evaluation inside the body of an uninstantiated function (which means *partial evaluation*). The choice of this weak version follows an approach by Blanc, Lévy and Maranget and is closely related to fully lazy sharing [BLM07]. Please note that this work considers optimality relative to the weak restriction, which differs from the usual strong theory of β-optimality mentioned above.

Correctness and optimality of a whole class of strategies for the graph implementation of *PPC* proposed in this paper are stated (Correctness & Optimality Corollary 5.7) by means of an axiomatic approach given by Glauert and Khasidashvili [GK96]. They provide abstract notions of *equivalent redexes* and *contribution to a redex* using a set of axioms which are sufficient to prove that some strategies turn out to be optimal.

Main difficulty for the user of such a technology is to exhibit concrete notions which are clever enough to satisfy all the axioms defining the abstract notions. The technique proposed here to construct satisfying notions of equivalence and contribution is a direct reuse of the labelling system that defines the graph implementation, which gives the axioms almost for free. This shows how the systematic analysis of history can base various advanced results and constructions on a system.

A second contribution of this work is to show on the example of PPC *an alternative method to derive optimality of needed strategies in concrete rewriting systems.*

Organization in short: Section 2 formally introduces the *Pure Pattern Calculus* and its weak restriction. Section 3 analyzes the ways different redexes can contribute to each other, and deduces a confluent labelled pattern calculus enjoying finite developments properties (Confluence Theorem 3.12 and Finite Developments Theorem 3.32). The labelled calculus is linked to a graph reduction system in Section 4 (through Preservation of Sharing Theorem 4.9). Section 5 finally states correctness and optimality of some reduction strategies (Correctness & Optimality Lemma 5.7). Related work is reviewed in Section 6, before a conclusion is drawn.

2 First-Class Patterns

The *Pure Pattern Calculus* (*PPC*) of Jay and Kesner [JK09] is a functional framework featuring data structures and pattern matching with path and pattern polymorphisms. One of its attributes is to make patterns first-class citizens: they can be given as arguments or returned as results. They can also be evaluated so that they are called **dynamic patterns**. Moreover, any term is allowed to be used as a pattern *a priori*, the pattern matching operation being responsible for dynamically rejecting *improper* uses. Here is a reminder of the calculus, followed by the definition of its weak restriction.

Syntax. Grammar for terms is associated with its subcategory of **matchable forms**: *stable* terms that are ready for matching.

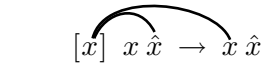
$$\begin{array}{rcll} a,b,p,r,s,t & ::= & x \mid \hat{x} \mid tt \mid [\theta]\, t \to t & \text{Terms} \\ d & ::= & \hat{x} \mid dt & \text{Data structures} \\ m & ::= & d \mid [\theta]\, t \to t & \text{Matchable forms} \end{array}$$

where $x, y, z \in \mathcal{X}$ the set of names, and θ, τ are lists of names.

The term $t_1 t_2$ is called an **application**, and $[\theta]\, p \to b$ a **case**. Letter p indicates a term used as ***p***attern, and b as a function ***b***ody. Letters a, r and s are also used below for ***a***rguments, ***r***edexes and ***s***ubterms. As usual, let $\mathcal{C}[]$ denote a context (term with a hole) and $\mathcal{C}[t]$ the context $\mathcal{C}[]$ where the hole is replaced by t. The natural notion of subterm of t at position $\mathfrak{p}$ is written $t|_{\mathfrak{p}}$.

Note that a name x has two kinds of occurrences: proper occurrences x as **variables** which can be substituted, and occurrences $\hat{x}$ as **matchables** which can't. The matchable $\hat{x}$ is a constant with a name x: it may be used either as a constructor or as a matching variable (which captures what will be substituted for x), depending on its free or bound status. Boldface symbol $\mathbf{c}$ denotes a ***c***onstructor in the examples.

Variable and matchable bindings. As pictured below, in the term $[\theta]\, p \to b$ the list θ binds matchables in p and variables in b, which corresponds to the following formal definitions for free variables $fv(t)$ and free matchables $fm(t)$ of a term t. **Free names** of a term t are defined as $fn(t) = fv(t) \cup fm(t)$.



$$\begin{array}{rcl} fv(x) & := & \{x\} \\ fv(\hat{x}) & := & \emptyset \\ fv(t_1 t_2) & := & fv(t_1) \cup fv(t_2) \\ fv([\theta]\, p \to b) & := & fv(p) \cup (fv(b) \setminus \theta) \end{array}$$

$$\begin{array}{rcl} fm(x) & := & \emptyset \\ fm(\hat{x}) & := & \{x\} \\ fm(t_1 t_2) & := & fm(t_1) \cup fm(t_2) \\ fm([\theta]\, p \to b) & := & (fm(p) \setminus \theta) \cup fm(b) \end{array}$$

A natural notion of α-conversion is deduced from these binding rules. For any $x \in \theta$ and y fresh:

$$[\theta]\, p \to b \quad =_\alpha \quad [\theta\{x := y\}]\, p\{\hat{x} := \hat{y}\} \to b\{x := y\}$$

For now on, bound names of any term will be considered all different (and also different from the free names).

Example 2.

$$[y]\, \hat{x}y\hat{y} \to xy\hat{y} =_\alpha [z]\, \hat{x}y\hat{z} \to xz\hat{y}$$

In the pattern $\hat{x}y\hat{y}$ the matchable $\hat{y}$ is bound and is used as a matching variable, whereas $\hat{x}$ is free and seen as a constructor. y is a free variable: an external parameter.

Substitution. Substitution, as in λ-calculus, is a meta-operation defined by equations. Notation $\theta\#\sigma$ (θ **avoids** σ) stands for $\theta \cap (dom(\sigma) \cup fn(cod(\sigma))) = \emptyset$, where $dom(\sigma)$ (resp. $cod(\sigma)$) denotes the domain (resp. codomain) of the substitution σ. For terms, $\theta\#t$ is $\theta \cap fn(t) = \emptyset$.

$$\begin{array}{rcll} x^\sigma & := & \sigma_x & x \in dom(\sigma) \\ x^\sigma & := & x & x \notin dom(\sigma) \\ \hat{x}^\sigma & := & \hat{x} & \\ (t_1 t_2)^\sigma & := & t_1^\sigma t_2^\sigma & \\ ([\theta]\, p \to b)^\sigma & := & [\theta]\, p^\sigma \to b^\sigma & \theta\#\sigma \end{array}$$

Pattern matching. The result of the pattern matching operation is called a **match** (meta-variable μ), and is either a substitution in case of successful matching, or the symbol $\bot$ for matching failure. The match of an argument a against a pattern p with matching variables θ is noted $\{a/[\theta]\,p\}$. Its definition is based on the following **compound matching** operation, where equations are to be taken in order:

$$\begin{array}{rcll}
\{\!\{a/[\theta]\,\hat{x}\}\!\} & := & \{x \mapsto a\} & x \in \theta \\
\{\!\{\hat{x}/[\theta]\,\hat{x}\}\!\} & := & \{\} & x \notin \theta \\
\{\!\{a_1a_2/[\theta]\,p_1p_2\}\!\} & := & \{\!\{a_1/[\theta]\,p_1\}\!\} \uplus \{\!\{a_2/[\theta]\,p_2\}\!\} & \\
 & & a_1a_2 \text{ and } p_1p_2 \text{ are matchable forms} & \\
\{\!\{a/[\theta]\,p\}\!\} & := & \bot & \\
 & & a \text{ and } p \text{ are matchable forms, otherwise} & \\
\{\!\{a/[\theta]\,p\}\!\} & := & \texttt{wait} & \text{otherwise}
\end{array}$$

where the $\uplus$ operator stands for disjoint union of substitutions. Union of matchs $\mu_1 \uplus \mu_2$ is $\bot$ if μ_1 or μ_2 is $\bot$ or if the domains of μ_1 and μ_2 overlap. The result `wait` is undefined and corresponds to the case where the pattern or the argument has still to be evaluated or instantiated.

Example 3.

Let $\boldsymbol{c}$ be a constructor. $\{\!\{\boldsymbol{c}/[y]\,x\hat{y}\}\!\}$ is undefined (`wait`) since the pattern $x\hat{y}$ starting with a variable is not a matchable form. $\{\!\{\boldsymbol{c}/[y]\,([z]\,\hat{z} \to z\boldsymbol{c})\hat{y}\}\!\}$ is also undefined: the pattern is now a preredex (definition below), which is not a matchable form either. The third attempt $\{\!\{\boldsymbol{c}/[y]\,\hat{y}\boldsymbol{c}\}\!\}$ is defined as $\bot$, since the atomic argument $\boldsymbol{c}$ do not match the compound pattern $\hat{y}\boldsymbol{c}$. An example of successful compound matching is $\{\!\{\boldsymbol{c}([z]\,\hat{z} \to z\boldsymbol{c})/[x_1x_2]\,\hat{x_1}\hat{x_2}\}\!\}$: the argument and the pattern are matchable forms, and each of the bound matchables $\hat{x_1}$ and $\hat{x_2}$ captures a part of the argument to yield the substitution $\{x_1 \mapsto \boldsymbol{c},\ x_2 \mapsto ([z]\,\hat{z} \to z\boldsymbol{c})\}$.

Now suppose $\{\!\{a/[\theta]\,p\}\!\} = \sigma$. If $\theta = dom(\sigma)$ then define $\{a/[\theta]\,p\} = \sigma$, else $\{a/[\theta]\,p\} = \bot$. This **check** operation ensures that the pattern p contains all the matchables whose names are bound by θ.

Reduction. As in λ-calculus, reduction is defined by one single rule (called β_m) which is still a meta-operation, performing pattern matching and substitution in one step. Any subterm of the form $r = ([\theta]\,p \to b)a$ is called a **preredex**. If $\{a/[\theta]\,p\}$ is defined (not `wait`) then the preredex r is a **redex** and the rule β_m applies. For any term b, define $b^\bot$ as some fixed closed normal form $\bot$.

$$([\theta]\,p \to b)a \quad \xrightarrow{r}_{\beta_m} \quad b^{\{a/[\theta]\,p\}}$$

The choice here, as in [JK09], is the identity: $\bot = [x]\,\hat{x} \to x$. This allows in particular to catch matching failures, and then to trigger alternative or default cases. The result $b^{\{a/[\theta]\,p\}}$ is called **contractum** of r. For now, the rule can be applied in any context. The reduction ρ of a redex r in a term t is noted $\rho : t \xrightarrow{r}_{\beta_m} t'$. Annotations β_m, r and ρ can be omitted when the information is not needed.

Reduction inside *any context* is formalized as follows:

$$\frac{t_1 \xrightarrow{r}_{\beta_m} t_1'}{t_1t_2 \xrightarrow{r}_{\beta_m} t_1't_2} \qquad \frac{t_2 \xrightarrow{r}_{\beta_m} t_2'}{t_1t_2 \xrightarrow{r}_{\beta_m} t_1t_2'} \qquad (\zeta)\ \frac{p \xrightarrow{r}_{\beta_m} p'}{[\theta]\,p \to b \xrightarrow{r}_{\beta_m} [\theta]\,p' \to b}$$

$$(\xi)\ \frac{b \xrightarrow{r}_{\beta_m} b'}{[\theta]\,p \to b \xrightarrow{r}_{\beta_m} [\theta]\,p \to b'}$$

Example 4.

RUNNING EXAMPLE. *Fragments of* Example 3 *are gathered here. Contracted* <u>redexes</u> *are underlined:*

$$\begin{array}{lll} \underline{([x_1x_2]\,\hat{x_1}\hat{x_2} \to ([y]\,x_2\hat{y} \to b)\boldsymbol{c})\ \ (\boldsymbol{c}([z]\,\hat{z} \to z\boldsymbol{c}))} & & (1)\\ \quad \to_{\beta_m} \quad ([y]\,\underline{([z]\,\hat{z} \to z\boldsymbol{c})\hat{y}} \to b)\boldsymbol{c} & & (2)\\ \quad \to_{\beta_m} \quad \underline{([y]\,\hat{y}\boldsymbol{c} \to b)\,\boldsymbol{c}} & & (3)\\ \quad \to_{\beta_m} \quad \bot & & (4) \end{array}$$

Symbol ρ is for one-step reduction, and $\overrightarrow{\rho}$ for a sequence. A **normal form** is a term which can not be further reduced. A term t is **normalizable** if there exists $\overrightarrow{\rho} : t \to t'$ with t' a normal form. A term t is **terminating** or strongly normalizing if there is no infinite reduction from t.

Remark on expressivity Note that λ-calculus enjoys a direct encoding into *PPC*: the λ-term $\lambda x.t$ can be rewritten as the *PPC*-term $[x]\,\hat{x} \to t$. To recover β-reduction $(\lambda x.t)u \to_\beta t^{\{x \mapsto u\}}$, remark that application $[x]\,\hat{x} \to t$ generates the trivial matching $\{u/[x]\,\hat{x}\}$ which results in the substitution $\{x \mapsto u\}$.

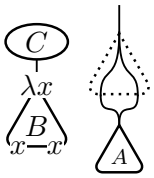
Weak Pure Pattern Calculus. At run-time, evaluation of functional programs is mainly a matter of passing arguments to functions, and not evaluating functions bodies before they get instantiated. This leads to the study of **weak reduction**, where the reduction rule is never applied *under an abstraction*. This reduction relation is formalized by removing the (ξ)-rule of *PPC*. Remark that the (ζ)-rule concerning pattern reduction is kept, even if it's in some way *under an abstraction*. The reasons are that in the pattern, only matchables (which can't be substituted) are bound and not variables, and that reduction of the pattern may be *necessary* when a case is applied to an argument!

As in λ-calculus, removing (ξ) breaks confluence. One solution preserving confluence [ÇH98] adopts a restricted rule which limits reduction under an abstraction to subterms independent of the abstracted variables. Replacing (ξ) by (ξ') defines ***WPPC***, the ***Weak Pure Pattern Calculus***.

$$(\xi')\ \frac{b \ \xrightarrow{r}_{\beta_m} \ b' \qquad \theta \# r}{[\theta]\,p \to b \quad \xrightarrow{r}_{\beta_m} \quad [\theta]\,p \to b'}$$

Now, given a preredex $r = ([\theta]\,p \to b)a$ in a term t, the definition of $\{a/[\theta]\,p\}$ is not enough for r to be a redex. It also requires r to be *closed in* t: variables free in r should not be bound outside. Remark that with the convention on names, condition $\theta \# r$ in (ξ') is equivalent to $\theta \cap fv(r) = \emptyset$.

Remark on the reduction under an abstraction. This definition of *WPPC* allows some reductions in the scope of an abstraction. This is required for full laziness. Indeed, consider some redex r in the dotted zone in the following picture (taken from the introduction). This redex is shared between B and the dotted zone, and hence can be reached by two paths: on the right its reduction can be required in order to reach a usual weak head normal form, whereas on the left it is seen as *under an abstraction*. And since these two versions of r are shared, they are reduced at the same time.



Descendants and residuals. For a reduction $\rho : t \xrightarrow{r}_{\beta_m} t'$ and a subterm s_a of t, **descendants** of s_a after ρ, noted s_a/ρ, are the subterms s_d of t' that *come from* s_a. If $s_d \in s_a/\rho$,

then call s_a an **ancestor** of s_d. Descendants after a one-step reduction are described in the enumeration below. Write $r = ([\theta]\, p \to b)a \ \to_{\beta_m} \ r'$.

1. If s_a is disjoint from r, then s_a remains unchanged, $s_a/\rho = \{s_a\}$.
2. If r is a strict subterm of s_a, that is $s_a = \mathcal{C}[r]$ with $\mathcal{C}[] \neq []$, then $s_a/\rho = \{\mathcal{C}[r']\}$.
3. If $\{a/[\theta]\, p\} = \sigma$ and s_a is a subterm of b but not a variable in θ, then $s_a/\rho = \{s_a^\sigma\}$.
4. If $\{a/[\theta]\, p\} = \sigma$ and s_a is a subterm of a which is in the codomain of σ, then let x be the variable and $\mathfrak{p}$ the position such that $\sigma_x|_{\mathfrak{p}} = s_a$. For each position $\mathfrak{q}$ such that $b|_{\mathfrak{q}} = x$, the subterm s_a has a residual at position $\mathfrak{q}\mathfrak{p}$ in the residual b^σ of b.
5. In any other case, s_a has no residual: $s_a/\rho = \emptyset$.

An extension to general reductions is given by:

- For any set $\mathcal{S}$ of subterms, consider all descendants: $\mathcal{S}/\rho = \{s_d \mid \exists s_a \in \mathcal{S}, s_d \in s_a/\rho\}$.
- For any non-empty sequence of reduction $\overrightarrow{\rho}$, and for ρ_0 a one-step reduction, $\mathcal{S}/(\overrightarrow{\rho}\,\rho_0) = (\mathcal{S}/\overrightarrow{\rho})/\rho_0$.

The term **residual** denotes a redex which is the descendant of a redex. A redex r_c is **created** by ρ if it is not the descendant of a redex (which doesn't mean that r_c has no ancestor).

Example 5.
In Running Example 4, $([z]\, \hat{z} \to z\boldsymbol{c})\hat{y}$ *in term* (2) *is a descendant (in fact, the unique descendant) of* $x_2\hat{y}$ *from term* (1). *It is also a redex created by this reduction step (and not a residual) since the ancestor* $x_2\hat{y}$ *was not a redex. On the other hand, the application* $\boldsymbol{c}([z]\, \hat{z} \to z\boldsymbol{c})$ *of term* (1) *is destroyed by the matching and has no descendant.*

Developments. Let t be a term, and $\mathcal{R}$ be a set of redexes of t. A **development** of $\mathcal{R}$ is a sequence of reduction $\overrightarrow{\rho} = \rho_1 ... \rho_n$ that contracts only (residuals of) redexes of $\mathcal{R}$: for any $i \in 2...n$, ρ_i contracts a redex in $\mathcal{R}/\rho_1 ... \rho_{i-1}$. A development $\overrightarrow{\rho}$ of $\mathcal{R}$ is **complete** when $\mathcal{R}$ has no descendant after $\overrightarrow{\rho}$: $\mathcal{R}/\overrightarrow{\rho} = \emptyset$.

WPPC is confluent, and enjoys traditional properties of finite developments. These facts are corollaries of Confluence Theorem 3.12 and Finite Developments Theorem 3.32 on the labelled calculus defined in Section 3.

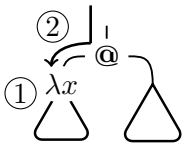
3 The Labelled Weak Pure Pattern Calculus

3.1 From Causality to Labels

One slogan of optimality theory [Lé80] is *redexes with same origin should never get separated, and should therefore be reduced in one unique step.* The notion of origin of a redex r is to be understood here as the set of all past reduction events that were *necessary* for r to be a redex. Formal words for this are: the reduction steps that **contribute** to the creation of the redex r.

This subsection analyzes this contribution relation, and prepares the derict recording of contribution into terms via labels. The first step here is to characterize the cases of direct contribution, which correspond to the cases where a redex can be created.

Creation of redexes in λ-calculus has been studied in [Lé78] and classified in three cases. For the purpose of this paper a coarser point of view is enough, and Lévy's classification is summed up in two cases:

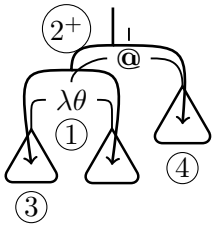


In λ-calculus this redex is created when the abstraction comes in touch with the application. This can be due either to reduction of the left part of the application into an abstraction (1) (which covers the two first cases of Lévy) or to an external substitution replacing some variable occurrence with the abstraction (2) (which is exactly the third case of Lévy). In the weak calculus studied here, a new case of creation arises: when the term has already the shape of a redex, but is not one due to an occurrence of an externally bound variable. The preredex becomes a redex when this occurrence is substituted. This is captured by a generalization of case (2) where a substitution acts anywhere inside the left or right part of the application, which is written (2^+) below.

In the following pictures, an abstraction $[\theta]\, p \to b$ is denoted by a binary node $\lambda\theta$ whose left son represents the pattern p and right son the body b, as below.

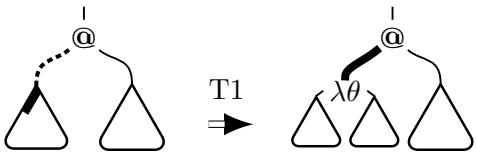
$$[\theta]\, p \to b \quad \equiv$$

Pattern matching and dynamic patterns bring two symmetrical new cases of creation. Indeed, when the abstraction of a pattern p is applied to an argument a then a has to be matched against p. But, as seen in the previous section, $\{a/[\theta]\, p\}$ is not always defined. In particular some parts of a or p may be required to be in matchable form while still having to be evaluated. Then the two new cases are: reduction in the pattern (3) (due to the use of dynamic patterns) and reduction in the argument (4) (which appears in any pattern matching frameworks). The other case of non-matchable form is the presence of a variable occurrence that has to be substituted and falls in the case (2^+) of substitution.



3.1.1 Redex Creation Theorem

The first case is the contraction of a redex which is left of an application and results in an abstraction.



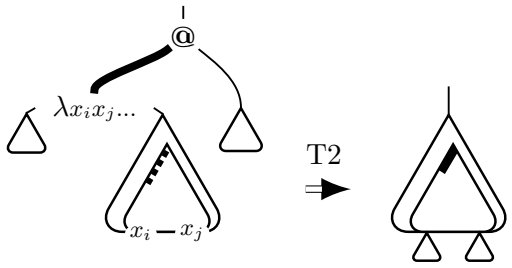
Example 6.

This creation type covers the two first cases of Lévy and the handling of failures:

$$((\![\emptyset]\, \hat{c} \to ([\theta]\, p \to b))\hat{c})a \;\to ([\theta]\, p \to b)a$$
$$((\![x]\, \hat{x} \to x)([\theta]\, p \to b))a \to ([\theta]\, p \to b)a$$

Failure: $((\![\emptyset]\, \hat{c} \to ([\theta]\, p \to b))\hat{c}\hat{c})a \;\to \bot a = ([x]\, \hat{x} \;\to\; x)a$

Second case is the instantiation of variables of the ancestor of the created redex:



Example 7.

Weak case: a preredex isn't a redex due to the only presence of some variables (ancestors of created redexes are underlined here).

$$([x]\,\hat{x} \to ((([y]\,\hat{y} \to xy)a))t \to ([y]\,\hat{y} \to ty)a$$

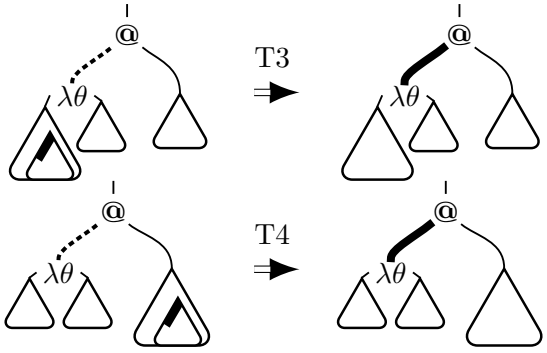
This also contains Lévy's third case:

$$([x]\,\hat{x} \to (xa))([y]\,\hat{y} \to b) \to ([y]\,\hat{y} \to b)a$$

And last, instantiation of a pattern is also covered here.

$$([x]\,\hat{x} \to ((([y]\,x\hat{y} \to b)\hat{c}a))\hat{c} \to ([y]\,\hat{c}\hat{y} \to b)\hat{c}a$$

Third and fourth cases: reduction in the pattern or in the argument.



Example 8.

In Running Example 4*, the redex of term* (2) *is created by the first step along* T2*, whereas the redex of term* (3) *is created by the second step along* T3*.*

Theorem 3.1 (Redex Creation). *Let* $\mathcal{C}[r] \xrightarrow{r} t'$*. Suppose* r_c *is a redex created by this reduction with* $t' = \mathcal{C}_c[r_c]$*. Write* $r = ([\theta]\,p \to b)a \to_{\beta_m} b^\mu = r'$ *and* $r_c = ([\theta_c]\,p_c \to b_c)a_c$*. One of the following holds:*

T1. $\mathcal{C}[] = \mathcal{C}_c[[]a_c]$ *and* $r' = [\theta_c]\,p_c \to b_c$*.*

T2. $\mathcal{C}_c[] = \mathcal{C}[\mathcal{C}_c^-[]]$*,* $b = \mathcal{C}_c^-[r_c^*]$*,* $fv(r_c^*) \cap \theta \neq \emptyset$*, and* $(r_c^*)^\mu = r_c$*.*

T3. $\mathcal{C}[] = \mathcal{C}_c[([\theta_c]\,\mathcal{C}^-[] \to b_c)a_c]$*,* $\{a_c/[\theta_c]\,\mathcal{C}^-[r]\} = \texttt{wait}$ *(and* $\mathcal{C}^-[r] = p_c^* \xrightarrow{r} p_c$*).*

T4. $\mathcal{C}[] = \mathcal{C}_c[([\theta_c]\,p_c \to b_c)\mathcal{C}^-[]]$*,* $\{\mathcal{C}^-[r]/[\theta_c]\,p_c\} = \texttt{wait}$ *(and* $\mathcal{C}^-[r] = a_c^* \xrightarrow{r} a_c$*).*

The case of creation of a redex at the root of a term is isolated in the following lemma:

Lemma 3.2. *Under the hypothesis of Theorem 3.1, if* $\mathcal{C}_c[] = []$ *and* $\mathcal{C}[] \neq []$*, then* T1*,* T3 *or* T4 *holds.*

Proof. First note that $\mathcal{C}_c[] = []$ implies $r_c = t' = \mathcal{C}[r']$. Now proof by case on $\mathcal{C}[] \neq []$:

- If $\mathcal{C}[] = [\theta_0]\,\mathcal{C}^-[] \to b_0$ or $\mathcal{C}[] = [\theta_0]\,p_0 \to \mathcal{C}^-[]$, there exists a pair p'_0, b'_0 such that $t' = [\theta_0]\,p'_0 \to b'_0$. t' isn't a preredex, hence $\mathcal{C}_c[]$ can't be the empty context $[]$.
- If $\mathcal{C}[] = t_0\mathcal{C}^-[]$, then $t_0\mathcal{C}^-[r'] = t' = r_c = ([\theta_c]\,p_c \to b_c)a_c$. Thus $t_0 = [\theta_c]\,p_c \to b_c$ and $\mathcal{C}^-[r'] = a_c$. Since $t = ([\theta_c]\,p_c \to b_c)\mathcal{C}^-[r]$ is not a redex (r_c is created), $\{\mathcal{C}^-[r]/[\theta_c]\,p_c\} =$ `wait`. T4 holds.
- If $\mathcal{C}[] = \mathcal{C}^-[]t_0$, then $t' = \mathcal{C}^-[r']t_0$ and $t_0 = a_c$. Case on $\mathcal{C}^-[]$:
  - If $\mathcal{C}^-[] = []$, then $\mathcal{C}[] = []a_c$ and $r_c = \mathcal{C}[r'] = r'a_c$. Then $r' = [\theta_c]\,p_c \to b_c$ and T1 holds.
  - If $\mathcal{C}^-[] = t_1[]$ or $\mathcal{C}^-[] = []t_2$, then there exists a pair t'_1, t'_2 such that $\mathcal{C}^-[r'] = t'_1t'_2$. Hence $\mathcal{C}^-[r'] \neq [\theta_c]\,p_c \to b_c$ and t' can't be equal to r_c.
  - If $\mathcal{C}^-[] = [\theta_0]\,p_0 \to \mathcal{C}^{--}[]$, then $r_c = t' = ([\theta_0]\,p_0 \to \mathcal{C}^{--}[r'])a_c$, $p_0 = p_c$ and $\theta_0 = \theta_c$. Since $t = ([\theta_c]\,p_c \to \mathcal{C}^{--}[r])a_c$ isn't a redex, $\{a_c/[\theta_c]\,p_c\} =$ `wait`. Hence r_c isn't a redex.
  - If $\mathcal{C}^-[] = [\theta_0]\,\mathcal{C}^{--}[] \to b_0$, then $r_c = t' = ([\theta_0]\,\mathcal{C}^{--}[r'] \to b_0)a_c$, $b_0 = b_c$ and $\theta_0 = \theta_c$. Since $t = ([\theta_c]\,\mathcal{C}^{--}[r] \to b_c)a_c$ isn't a redex, $\{a_c/[\theta_c]\,\mathcal{C}^{--}[r]\} =$ `wait`, and T3 holds. □

Proof of Redex Creation Theorem 3.1. Case on relative positions of r' and r_c in t'.

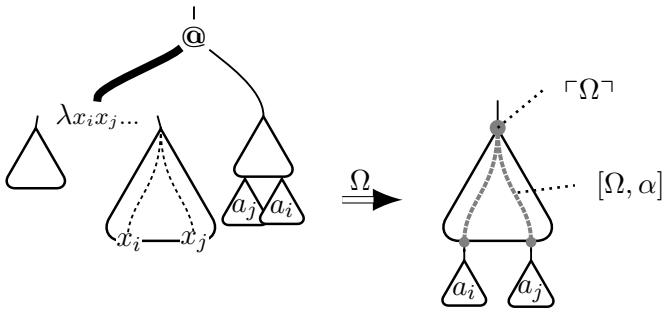
- If r' and r_c are disjoint, then r_c already exists in t and isn't created by r.
- If r' is a subexpression of r_c. Write $t = \mathcal{C}_c[r^*_c] = \mathcal{C}_c[\mathcal{C}^-[r]]$. Then $r^*_c = \mathcal{C}^-[r]$ is not a redex and is turned into one by reduction of its subterm r. Lemma 3.2 applies.
- If r_c is a subexpression of $r' = b^\mu$. Remark in this case that μ is a substitution σ (else $r' = \bot$, which is a normal term). Suppose r_c is in the codomain of σ. Then r_c was a subterm of a, and was already a redex. Hence r_c is rooted in b: $\mathcal{C}_c[] = \mathcal{C}[\mathcal{C}^-_c[]]$. The ancestor of r_c is r^*_c such that $b = \mathcal{C}^-_c[r^*_c]$ and $(r^*_c)^\sigma = r_c$.
  - If $\theta_c \cap fv(r^*_c) = \emptyset$, then $r_c = r^*_c$, and r^*_c was already a redex.
  - If $\theta_c \cap fv(r^*_c) \neq \emptyset$, then T2 holds. □

A labelling system can be constructed by inverting the above description of redex creation: the question is not anymore *where does this redex come from?* but *what can be the future consequences of this reduction step?* The approach turns from *backward* to *forward*. If each reduced redex drops its fingerprint wherever it can possibly contribute to something (but nowhere else!), then the origin of a redex can be known by collecting the information left at this place by past reductions.

To achieve this, labels are added to the syntax of *WPPC*: every subterm bears a label recording any locally relevant information about past reductions, and reduction acts on these labels to keep contribution information up to date. For this, each redex gets a name deduced from neighbouring labels (in a way precised in Section 3.2), name which is used to track the future contributions of the redex. In such a framework the optimality commandment becomes more concrete by switching to: *reduce in one step all redexes with same name.*

The reduction of a redex r of name Ω transforms the labels of its contractum. Firstly r can contribute to something at its root through case (1): a label $\ulcorner\Omega\urcorner$ (denoting the *epicentre of* Ω) is added at the root of the contractum to witness this. Secondly if r is a successful match, then it generates a substitution. In this case r can contribute to something through

case (2^+) anywhere along the propagation of the substitution. This defines a connected area of the contractum which is referred to in this paper as **substitution slice** (it is a slice in the sense of [Ter03, Chap. 8]). The substitution slice is the union of the paths from the root of the contractum to each substituted variable occurrence. This other kind of contribution is witnessed in the labelling by an other construct: each atomic label α in the substitution slice is turned into another atomic label $[\Omega, \alpha]$ which can be understood as *a copy of* α *triggered by* Ω. This is summed up in the following picture.



Contributions from pattern or argument reduction are not visible here: indeed these contributions do not concern the contractum of the redex, but some other undetermined places in the context. The design of a mechanism taking into account these non local contributions is the main difficulty of the next section. To achieve this, the *forward* labelling aspects presented above are mixed with a *backward* analysis of the pattern and the argument: any relevant information found in the pattern or the argument of a redex is included in its name. The difficulty is then to discriminate between relevant and irrelevant parts of the pattern and the argument: they are not the same in case of success or failure of the pattern matching. Thus relevant prefixes of the pattern and the argument are mutually dependant and have to be dynamically determined by the pattern matching operation itself.

3.2 Formalizing Labels

This subsection defines the *Labelled Weak Pure Pattern Calculus* (*LWPPC*), which embeds a characterization of the aforementioned contribution relation.

Syntax of *PPC* is extended with the labels introduced above. In the grammar, terms that must (resp. that must not) have a label in root position are called labelled (resp. clipped).

$$
\begin{array}{rcll}
A, B, P, R, S, T & ::= & \alpha : X & \text{Labelled terms} \\
X, Y, Z & ::= & T \mid N & \text{Terms} \\
N & ::= & x \mid \hat{x} \mid TT \mid [\theta]\, T \to T & \text{Clipped terms} \\
\alpha & ::= & \mathbb{\alpha} \mid \ulcorner\Omega\urcorner \mid [\Omega, \alpha] & \text{Atomic labels} \\
\Gamma, \Delta, \Omega & ::= & \alpha_1 \alpha_2 \ ... \ \alpha_n & \text{Labels}
\end{array}
$$

where n is a strictly positive integer, $x, y, z \in \mathcal{X}$ the set of names and $\mathbb{\alpha}, \mathbb{\beta} ... \in \mathbb{A}$ the set of initial labels (blackboard bold greek letters are used to distinguish initial labels when needed). An **initial term** is a term whose labels are all initial and different.

For any possibly non-atomic label $\Gamma = \alpha_1 \ ... \ \alpha_n$, write $\Gamma \cdot X$ as a shorthand for $\alpha_1 : ... : \alpha_n : X$. Letters P, B and A are still used for terms playing the role of pattern, function body and argument, while the greek letter Ω indicates the name of a redex.

Formal definition of the set of positions $\mathcal{P}(X)$ of a term X, is defined as follows.

$$\begin{array}{rcl}
\mathcal{P}(x) & := & \{\epsilon\} \\
\mathcal{P}(\hat{x}) & := & \{\epsilon\} \\
\mathcal{P}(T_1T_2) & := & \{\epsilon\} \cup \mathbb{1}\mathcal{P}(T_1) \cup \mathbb{2}\mathcal{P}(T_2) \\
\mathcal{P}([\theta]\, P \to B) & := & \{\epsilon\} \cup \mathbb{p}\mathcal{P}(P) \cup \mathbb{b}\mathcal{P}(B) \\
\mathcal{P}(\alpha : Z) & := & \{\epsilon\} \cup \mathbb{z}\mathcal{P}(Z)
\end{array}$$

The subterm of X at position $\mathfrak{p}$ is noted $X|_{\mathfrak{p}}$. The term X with subterm at position $\mathfrak{p}$ replaced by Y is noted $X[Y]_{\mathfrak{p}}$. Note that positions of atomic labels are taken into account.

Notions of free variables, free matchables, and substitution inherited from *PPC* are detailed here:

$$\begin{array}{rcl@{\qquad}rcl}
fm(x) & := & \emptyset & fv(x) & := & \{x\} \\
fm(\hat{x}) & := & \{x\} & fv(\hat{x}) & := & \emptyset \\
fm(T_1T_2) & := & fm(T_1) \cup fm(T_2) & fv(T_1T_2) & := & fv(T_1) \cup fv(T_2) \\
fm([\theta]\, P \to B) & := & (fm(P) \setminus \theta) \cup fm(B) & fv([\theta]\, P \to B) & := & fv(P) \cup (fv(B) \setminus \theta) \\
fm(\alpha : Z) & := & fm(Z) & fv(\alpha : Z) & := & fv(Z)
\end{array}$$

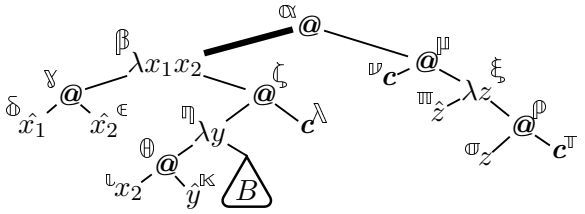
$$fn(X) = fm(X) \cup fv(X)$$

$$\begin{array}{rcll}
x^\sigma & := & \sigma_x & x \in dom(\sigma) \\
x^\sigma & := & x & x \notin dom(\sigma) \\
\hat{x}^\sigma & := & \hat{x} & \\
(T_1T_2)^\sigma & := & T_1^\sigma T_2^\sigma & \\
([\theta]\, P \to B)^\sigma & := & [\theta]\, P^\sigma \to B^\sigma & \theta \# \sigma \\
(\alpha : Z)^\sigma & := & \alpha : Z^\sigma &
\end{array}$$

For σ and σ' two substitution with same domain θ, write $\sigma \xrightarrow{R} \sigma'$ when there is $x \in \theta$ such that $\sigma_x \xrightarrow{R} \sigma'_x$, and $\sigma_y = \sigma'_y$ for all $y \in \theta \setminus \{x\}$.

Example 9.

Term (1) *of* Running Example 4 *with an initial labelling (referred to as* (L1) *in future examples).*



The **direct contribution** $\prec$ is a well-founded relation over labels which turns out to be useful in following sections. The notion of direct contribution appears in [Mar92] for term rewriting systems. It is adapted for *LWPPC* as follows: $\Omega \prec \Gamma$ if and only if $\Gamma = \Gamma_1 \ulcorner \Omega \urcorner \Gamma_2$ or $\Gamma = \Gamma_1 [\Omega, \alpha] \Gamma_2$ (both Γ_1 and Γ_2 may be empty here).

The following grammar extends notions of data structres and matchables forms to labelled terms.

$$\begin{array}{lcll}
D_l & ::= & \alpha : D & \text{Labelled data structures} \\
D & ::= & \hat{x} \mid D_l T \mid D_l & \text{Data structures} \\
M & ::= & D \mid [\theta]\, P \to B \mid \alpha : M & \text{Matchable forms}
\end{array}$$

To any matchable form M is associated a label $|M|$ called **matchability witness** which is meant to record past events that contributed to put M in matchable form.

$$\begin{array}{rcl} |[\theta]\, P \to B| & := & \varepsilon \\ |\hat{x}| & := & \varepsilon \\ |T_1 T_2| & := & |T_1| \\ |\alpha : Z| & := & \alpha|Z| \end{array}$$

The labelled compound matching returns the pair of a label Δ and a match μ. The label Δ is meant to collect on-the-fly any contribution information relative to the evaluation of the pattern and/or the argument.

For correct treatment of matchability witnesses, the labelled compound matching has two policies: **simple** compound matching $\{\!\{Y/[\theta]\, X\}\!\}_s$ records all the labels inside Y and X that are visited during the matching operation, whereas **witnessing** compound matching $\{\!\{Y/[\theta]\, X\}\!\}_w$ also records the matchability witness of Y even if Y isn't completely visited. In $\{\!\{Y/[\theta]\, X\}\!\}_p$ the subscript p is either s or w. Notation $\alpha : (\Delta, \mu)$ stands for $(\alpha\Delta, \mu)$.

$$\begin{array}{rcll} \{\!\{Y/[\theta]\, \alpha : Z\}\!\}_p & := & \alpha : \{\!\{Y/[\theta]\, Z\}\!\}_p & \\ \{\!\{Y/[\theta]\, \hat{x}\}\!\}_s & := & (\varepsilon, \{x \mapsto Y\}) & x \in \theta \\ \{\!\{Y/[\theta]\, \hat{x}\}\!\}_w & := & (|Y|, \{x \mapsto Y\}) & x \in \theta \\ \{\!\{Y/[\theta]\, ([\tau]\, P \to B)\}\!\}_s & := & (\varepsilon, \bot) & \\ \{\!\{Y/[\theta]\, ([\tau]\, P \to B)\}\!\}_w & := & (|Y|, \bot) & \\ \{\!\{\alpha : Z/[\theta]\, X\}\!\}_p & := & \alpha : \{\!\{Z/[\theta]\, X\}\!\}_p & \\ & & X \text{ matchable form, otherwise} & \\ \{\!\{\hat{x}/[\theta]\, \hat{x}\}\!\}_p & := & (\varepsilon, \{\}) & x \notin \theta \\ \{\!\{A_1 A_2/[\theta]\, P_1 P_2\}\!\}_p & := & \{\!\{A_1/[\theta]\, P_1\}\!\}_w \Subset \{\!\{A_2/[\theta]\, P_2\}\!\}_s & \\ & & A_1 A_2,\ P_1 P_2 \text{ matchable forms} & \\ \{\!\{Y/[\theta]\, X\}\!\}_p & := & (|Y||X|, \bot) & \\ & & Y, X \text{ matchable forms, otherwise} & \\ \{\!\{Y/[\theta]\, X\}\!\}_p & := & \texttt{wait} & \text{otherwise} \end{array}$$

where $\Subset$ is defined by the following table (with $\uplus$ the usual disjoint union of substitutions).

$\Subset$	(Δ_2, σ_2)	$(\Delta_2, \bot)$	`wait`
(Δ_1, σ_1)	$(\Delta_1\Delta_2, \sigma_1 \uplus \sigma_2)$	$(\Delta_1\Delta_2, \bot)$	`wait`
$(\Delta_1, \bot)$	$(\Delta_1, \bot)$	$(\Delta_1, \bot)$	$(\Delta_1, \bot)$
`wait`	`wait`	`wait`	`wait`

This operation is asymmetrical, due to the following difficulty: if an argument is incompatible with the pattern for two distinct reasons, which one should be blamed for it? In *PPC* any choice is fine, but with labelled terms distinct culprits yield distinct labels, and confluence is broken. This fact can be understood from a higher point of view by noticing that labelling of non-orthogonal calculi may lead to non-confluence [Ter03, Chap. 8]. And as pointed out in [KvOdV08], allowing matching failures as it is done in *PPC* breaks orthogonality (although the system still seems weakly orthogonal).

Remembering both failures is an easy but unsatisfying way to preserve confluence: it forces the matching operation to explore completely the pattern even if a failure has been detected, which doesn't seem to be a great solution when looking for an efficient evaluation model. The solution preferred here for solving this tricky problem is to set an order for pattern matching

operations (the left-to-right definition of $\Colon$ is just an example). Unfortunately, this causes a restriction on allowed reduction strategies. Any deterministic order being acceptable for each pattern matching, interesting degrees of freedom are preserved, but the way this could be used for optimization is unclear.

Remark that $\{\!\{Y/[\theta]\,X\}\!\}_w$ (with witnessing policy) is ill-defined if Y isn't a matchable form (in this case $|Y|$ has no sense). But if starting with simple policy, this case never happens.

Example 10.
For the redex in labelled term (L1) *(*Example 9*), the simple compound matching succeeds and returns the substitution* $\{x_1 \mapsto \mathbb{v} : \boldsymbol{c},\ x_2 \mapsto \mathbb{\xi} : ([z]\,\mathbb{m} : \hat{z} \to \mathbb{p} : (\mathbb{w} : z)(\mathbb{t} : \boldsymbol{c}))\}$ *and the label* $\Delta = \mathbb{y}\mathbb{l}\ \mathbb{d}\mathbb{v}\ \mathbb{e}$.

Given a label Ω and a list θ of names, the **relabelling** $\Omega(\!|\theta|\!)X$ of X is defined as X if $\theta\#X$ or X is a variable. Otherwise:

$$\begin{array}{rcl} \Omega(\!|\theta|\!)(T_1T_2) & := & (\Omega(\!|\theta|\!)T_1)(\Omega(\!|\theta|\!)T_2) \\ \Omega(\!|\theta|\!)([\tau]\,P \to B) & := & [\tau]\,(\Omega(\!|\theta|\!)P) \to (\Omega(\!|\theta|\!)B) \quad \theta \cap \tau = \emptyset \\ \Omega(\!|\theta|\!)(\alpha : Z) & := & [\Omega, \alpha] : (\Omega(\!|\theta|\!)Z) \end{array}$$

A **labelled preredex** is a labelled term of the form

$$R = \alpha : ((\Gamma \cdot [\theta]\,P \to B)A)$$

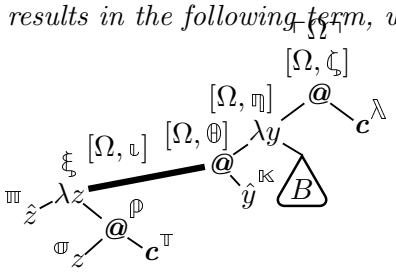
Define $\{A/[\theta]\,P\}$ to be the check of $\{\!\{A/[\theta]\,P\}\!\}_s$ against θ (default policy is the simple one). R is a **labelled redex** when it satisfies the restriction for weak reduction and $\{A/[\theta]\,P\} = (\Delta, \mu)$. Then the redex R get the name $\Omega = \alpha\Gamma\Delta$, and the rule is:

$$R \quad \xrightarrow{R} \quad \ulcorner\Omega\urcorner : (\Omega(\!|\theta|\!)B)^{\mu}$$

where as previously $X^{\perp} = \perp$ for any term X, with $\perp$-as-a-term a fixed closed normal form, for instance the identity with some distinguished initial labels:

$$\perp \quad = \quad [x]\,(\mathbb{m}_{\perp} : \hat{x}) \to (\mathbb{\beta}_{\perp} : x)$$

Example 11.
Reduction of (L1) *of Example* 9 *results in the following term, where* $\Omega = \mathbb{a}\ \mathbb{\beta}\ \mathbb{y}\mathbb{l}\mathbb{d}\mathbb{v}\mathbb{e}$



For any set $\mathcal{N}$ of labels, let $\to_{\mathcal{N}}$ be the reduction relation restricted to redexes with names in $\mathcal{N}$.

$$\frac{Z \ \xrightarrow{R}_{\mathcal{N}} \ Z'}{\alpha : Z \ \xrightarrow{R}_{\mathcal{N}} \ \alpha : Z'}$$

$$\frac{T_1 \ \xrightarrow{R}_{\mathcal{N}} \ T_1'}{T_1T_2 \ \xrightarrow{R}_{\mathcal{N}} \ T_1'T_2} \qquad \frac{T_2 \ \xrightarrow{R}_{\mathcal{N}} \ T_2'}{T_1T_2 \ \xrightarrow{R}_{\mathcal{N}} \ T_1T_2'} \qquad \frac{P \ \xrightarrow{R}_{\mathcal{N}} \ P'}{[\theta]\,P \to B \quad \xrightarrow{R}_{\mathcal{N}} \quad [\theta]\,P' \to B}$$

$$\frac{B \ \xrightarrow{R}_{\mathcal{N}} \ B' \qquad \theta\#R}{[\theta]\,P \to B \quad \xrightarrow{R}_{\mathcal{N}} \quad [\theta]\,P \to B'}$$

Notion of **residual**: for a reduction $\rho : X \xrightarrow{R}_{\mathcal{N}} X'$ and a subterm Y of X, with $R = \alpha : (\Gamma \cdot ([\theta]\, P \to B))A \to R'$, Y/ρ is described as:

- If Y is disjoint from R, Y remains unchanged, and is the unique residual of Y in X'.
- If R is a strict subterm of Y, that is $Y = \mathcal{C}[R]$ with $\mathcal{C}[] \neq []$, then the only residual of Y is $\mathcal{C}[R']$.
- If $\{A/[\theta]\, P\} = (\Delta, \sigma)$ and Y is a subterm of B but not a variable of θ, then the only residual of Y is $(\alpha\Gamma\Delta(\!|\theta|\!)Y)^\sigma$.
- If $\{A/[\theta]\, P\} = (\Delta, \sigma)$ and Y is a subterm of A which is in the codomain of σ. Note x the variable and $\mathfrak{p}$ the position such that $\sigma_x|_{\mathfrak{p}} = Y$. For each position $\mathfrak{q}$ such that $B|_{\mathfrak{q}} = x$, Y has a residual at position $\mathfrak{q}\mathfrak{p}$ in the residual $(\alpha\Gamma\Delta(\!|\theta|\!)B)^\sigma$ of B.
- In any other case, Y has no residual.

Remark there are more subterms without ancestors in *LWPPC*: when $R = \alpha : (\Gamma \cdot ([\theta]\, P \to B))A \to \ulcorner\Omega\urcorner : (\Omega(\!|\theta|\!)B)^\sigma = R'$, $(\Omega(\!|\theta|\!)B)^\sigma$ is a residual of B, but R' itself has no ancestor, due to the created label $\ulcorner\Omega\urcorner$.

The same notion of development is deduced: for X a term, and $\mathcal{R}$ a set of redexes of X. A **development** of $\mathcal{R}$ is a sequence of reduction $\overrightarrow{\rho} = \rho_1 ... \rho_n$ such that for any i, the redex contracted by ρ_i is in $\mathcal{R}/\rho_1 ... \rho_{i-1}$. A development $\overrightarrow{\rho}$ of $\mathcal{R}$ is **complete** when there is no residual of $\mathcal{R}$ left in the resulting term: $\mathcal{R}/\overrightarrow{\rho} = \emptyset$.

Remark that there are new subterms without ancestors in *LWPPC*: if $R = \alpha : (\Gamma \cdot ([\theta]\, P \to B))A \to \ulcorner\Omega\urcorner : (\Omega(\!|\theta|\!)B)^\sigma = R'$, then $(\Omega(\!|\theta|\!)B)^\sigma$ is a descendant of B, but R' itself has no ancestor, due to the created label $\ulcorner\Omega\urcorner$.

3.2.1 Labelled Creation Theorem

Theorem 3.3. *Let $\mathcal{C}[R] = X \xrightarrow{R}_{\mathcal{N}} X' = \mathcal{C}_c[R_c]$. Suppose R_c is a redex created by this reduction. Note $R = \alpha : ((\Gamma \cdot [\theta]\, P \to B)A) \to_{\mathcal{N}} R'$ and $R_c = \alpha_c : ((\Gamma_c \cdot [\theta_c]\, P_c \to B_c)A_c)$. Then one of the following cases holds.*

T1. $\mathcal{C}[] = \mathcal{C}_c[\alpha_c : ((\Gamma_c^1 \cdot [])A_c)]$, $R' = \Gamma_c^2 \cdot [\theta_c]\, P_c \to B_c$ *and* $\Gamma_c^1\Gamma_c^2 = \Gamma_c$ *(Γ_c^1 may be empty).*

T2. $\mathcal{C}_c[] = \mathcal{C}[\mathcal{C}_c^-[]]$, $B = \mathcal{C}_c^-[R_c^*]$, $fv(R_c^*) \cap \theta \neq \emptyset$, $\{A/[\theta]\, P\} = (\Delta, \sigma)$ *and* $(\alpha\Gamma\Delta(\!|\theta|\!)R_c^*)^\sigma = R_c$.

T3. $\mathcal{C}[] = \mathcal{C}_c[\alpha_c : (\Gamma_c \cdot ([\theta_c]\, \mathcal{C}^-[] \to B_c)A_c)]$, $\{A_c/[\theta_c]\, \mathcal{C}^-[R]\} = \texttt{wait}$ *($\mathcal{C}^-[R] = P_c^* \xrightarrow{R} P_c$).*

T4. $\mathcal{C}[] = \mathcal{C}_c[\alpha_c : (\Gamma_c \cdot ([\theta_c]\, P_c \to B_c)\mathcal{C}^-[])]$, $\{\mathcal{C}^-[R]/[\theta_c]\, P_c\} = \texttt{wait}$ *($\mathcal{C}^-[R] = A_c^* \xrightarrow{R} A_c$).*

The case of creation of a redex at the root of a term is isolated in the following lemma:

Lemma 3.4. *Under the hypothesis of Labelled Creation Theorem 3.3, if $\mathcal{C}_c[] = []$ and $\mathcal{C}[] \neq []$, then* T1*,* T3 *or* T4 *holds.*

Proof. If X is a clipped term, then X' is also a clipped term, and thus X' is not a labelled redex. Hence $X = \alpha_0 : Z$, and since $\mathcal{C}[] \neq []$, $X' = \alpha_0 : Z'$ with $Z \xrightarrow{R}_{\mathcal{N}} Z'$. Case on $\mathcal{C}[] \neq []$:

- If $\mathcal{C}[] = \alpha_0 : []$, then $Z \xrightarrow{Z}_{\mathcal{N}} Z'$ and Z' is labelled. Thus X' is not a labelled preredex.
- If $\mathcal{C}[] = \alpha_0 : \beta_0 : \mathcal{C}^-[]$, then $X' = \alpha_0 : \beta_0 : \mathcal{C}^-[R']$ and X' is not a labelled preredex.

- If $\mathcal{C}[] = \alpha_0 : [\theta_0]\,\mathcal{C}^-[] \to B_0$ or $\mathcal{C}[] = \alpha_0 : [\theta_0]\,P_0 \to \mathcal{C}^-[]$, there exists a pair P_0, B_0 such that $X' = \alpha_0 : [\theta_0]\,P_0 \to B_0$. X' isn't a preredex, hence $\mathcal{C}_c[]$ can't be the empty context $[]$.

- If $\mathcal{C}[] = \alpha_0 : (T_0\mathcal{C}^-[])$, then $\alpha_0 : (T_0\mathcal{C}^-[R']) = X' = \mathcal{C}_c[R_c] = R_c = \alpha_c : ((\Gamma_c \cdot [\theta_c]\,P_c \to B_c)A_c)$. Thus $\alpha_0 = \alpha_c$, $T_0 = \Gamma_c \cdot [\theta_c]\,P_c \to B_c$ and $\mathcal{C}^-[R'] = A_c$. Since $X = \alpha_c : ((\Gamma_c \cdot [\theta_c]\,P_c \to B_c)\mathcal{C}^-[R])$ is not a redex (R_c is created), $\{\mathcal{C}^-[R]/[\theta_c]\,P_c\} = \texttt{wait}$. T4 holds.

- If $\mathcal{C}[] = \alpha_0 : (\mathcal{C}^-[]T_0)$, then $X' = \alpha_0 : (\mathcal{C}^-[R']T_0)$ and $T_0 = A_c$. Case on $\mathcal{C}^-[]$:

  - If $\mathcal{C}^-[] = \Gamma_0 \cdot []$, then $\mathcal{C}[] = \alpha : ((\Gamma_0 \cdot [])A_c) = \mathcal{C}_c[\alpha_c : ((\Gamma_0 \cdot [])A_c)]$ and $R_c = X' = \mathcal{C}[R'] = \alpha : ((\Gamma_0 \cdot R')A_c)$. Then there is Γ_1 such that $R' = \Gamma_1 \cdot [\theta_c]\,P_c \to B_c$ and $\Gamma_0\Gamma_1 = \Gamma_c$, and T1 holds.
  - If $\mathcal{C}^-[] = \Gamma_0 \cdot (T_1[])$ or $\mathcal{C}^-[] = \Gamma_0 \cdot ([]T_2)$, then there exists a triple Γ_0, T_1, T_2 such that $\mathcal{C}^-[R'] = \Gamma_0 \cdot (T_1T_2) \neq \Gamma_c \cdot [\theta_c]\,P_c \to B_c$ and X' can't be equal to R_c.
  - If $\mathcal{C}^-[] = \Gamma_0 \cdot [\theta_0]\,P_0 \to \mathcal{C}^{--}[]$, then $R_c = X' = \alpha_0 : ((\Gamma_0 \cdot [\theta_0]\,P_0 \to \mathcal{C}^{--}[R'])A_c)$, $\alpha_0 = \alpha_c$, $\Gamma_0 = \Gamma_c$, $P_0 = P_c$ and $\theta_0 = \theta_c$. Since $X = \alpha_c : ((\Gamma_c \cdot [\theta_c]\,P_c \to \mathcal{C}^{--}[R])A_c)$ isn't a redex, $\{A_c/[\theta_c]\,P_c\} = \texttt{wait}$. Hence R_c isn't a redex.
  - If $\mathcal{C}^-[] = \Gamma_0 \cdot [\theta_0]\,\mathcal{C}^{--}[] \to B_0$, then $R_c = X' = \alpha_0 : ((\Gamma_0 \cdot [\theta_0]\,\mathcal{C}^{--}[R'] \to B_0)A_c)$, $\alpha_0 = \alpha_c$, $\Gamma_0 = \Gamma_c$, $b_0 = b_c$ and $\theta_0 = \theta_c$. Since $X = \alpha_c : ((\Gamma_c \cdot [\theta_c]\,\mathcal{C}^{--}[R] \to B_c)A_c)$ isn't a redex, $\{A_c/[\theta_c]\,\mathcal{C}^{--}[R]\} = \texttt{wait}$, and T3 holds. □

Proof of Labelled Creation Theorem 3.3. Case on relative positions of R' and R_c in X'.

- If R' and R_c are disjoint, then R_c already exists in X and isn't created by R.

- If R' is a subexpression of R_c. Note $X = \mathcal{C}_c[R_c^*] = \mathcal{C}_c[\mathcal{C}^-[R]]$. Then $R_c^* = \mathcal{C}^-[R]$ is not a redex and is turned into one by reduction of its subterm R. The Lemma 3.4 applies.

- If R_c is a subexpression of R'. Suppose $\{A/[\theta]\,P\} = (\Delta, \bot)$. Then $R' = (\alpha\Gamma\Delta(\!|\theta|\!)B)^{\bot} = \bot$ and R' is a normal form, which is not the case. Then $\{A/[\theta]\,P\} = (\Delta, \sigma)$ and $R' = (\alpha\Gamma\Delta(\!|\theta|\!)B)^{\sigma}$. Suppose R_c is in σ. Then R_c was a subterm of A, and was already a redex. Hence R_c is rooted in B: $\mathcal{C}_c[] = \mathcal{C}[\mathcal{C}_c^-[]]$. The ancestor of R_c is R_c^* such that $B = \mathcal{C}_c^-[R_c^*]$ and $(\alpha\Gamma\Delta(\!|\theta|\!)R_c^*)^{\sigma} = R_c$.

  - If $\theta_c \cap fv(R_c^*) = \emptyset$, then $R_c = R_c^*$, and R_c^* was already a redex.
  - If $\theta_c \cap fv(R_c^*) \neq \emptyset$, then T2 holds. □

3.3 Properties of LWPPC

3.3.1 Miscellaneous Lemmas

Lemma 3.5. *For any reduction $\rho : X \xrightarrow{R}_{\mathcal{N}} X'$ and subterm $\alpha : Z$ of X, residuals of $\alpha : Z$ after ρ have one of the following forms:*

- $\alpha : Z'$.

- $[\Omega, \alpha] : Z'$, *where Ω is the name of R.*

Proof. Case inspection on the definition of residuals. □

Lemma 3.6. *For any reduction $X \xrightarrow{R}_{\mathcal{N}} X'$ and labelled subterm $\alpha : Z'$ of X', the following holds:*

- *If α is an initial label, $\alpha : Z'$ is a residual of a subterm $\alpha : Z$ of X (or it may be created, but the latter is possible only for special labels of $\bot$).*
- *If $\alpha = \ulcorner\Omega\urcorner$, $\alpha : Z'$ is either created or a residual of a subterm $\alpha : Z$ of X.*
- *If $\alpha = [\Omega, \beta]$, $\alpha : Z'$ is a residual of a subterm of X of shape $\alpha : Z$ or $\beta : Z$.*

Proof. Case inspection on the definition of residuals. □

Lemma 3.7. *For any reduction $X \xrightarrow{R}_{\mathcal{N}} X'$ and labelled subterm $\alpha : N$ of X with N a clipped term. Residuals of N are exactly the clipped subterms of X' whose positions $\mathfrak{p}$ satisfy $\mathfrak{p} = \mathfrak{q}\mathbb{z}$ with $X'|_{\mathfrak{q}}$ a residual of $\alpha : N$.*

Proof. Note R' the contractum of the redex $R = \beta : (\Gamma \cdot ([\theta]\, P \to B)A)$. By case on the definition of residuals:

- $\alpha : N$ and R are disjoint if and only if N and R are disjoint. In this case the unique residual of $\alpha : N$ is itself, and the unique residual of N is also itself.
- If R is a strict subterm of $\alpha : N$ then R is a subterm of N. Since N is clipped and R is labelled, $N \neq R$ and R is a strict subterm of N: $N = \mathcal{C}[R]$ with $\mathcal{C} \neq []$ (conversely, any strict subterm of N is a strict subterm of $\alpha : N$). Thus the unique residual of $\alpha : N$ is $\alpha : \mathcal{C}[R']$, and the unique residual of N is $\mathcal{C}[R']$.
- If $\{A/[\theta]\, P\} = (\Delta, \sigma)$ and $\alpha : N$ is a subterm of B ($\alpha : N$ is not a variable), then the only residual of $\alpha : N$ is $(\beta\Gamma\Delta(\!|\theta|\!)\alpha : N)^\sigma = [\beta\Gamma\Delta, \alpha] : (\beta\Gamma\Delta(\!|\theta|\!)N)^\sigma$. Case on N (which is also a subterm of B):
  - If N is not a variable of θ, then its only residual is $(\beta\Gamma\Delta(\!|\theta|\!)N)^\sigma$.
  - If N is a variable x of θ, then it has no residual. But remark that in this case, the unique residual of $\alpha : N$ is $[\beta\Gamma\Delta, \alpha] : \sigma_x$ where σ_x is a labelled term (not clipped !).

  Remark that if N is a subterm of B, since N is clipped and B is labelled $N \neq B$ and N is a strict subterm of B. Thus $\alpha : N$ is also a subterm of B in this case.
- If $\{A/[\theta]\, P\} = (\Delta, \sigma)$ and $\alpha : N$ is a subterm of A which is in the codomain of σ, there is an x such that $\alpha : N$ is a subterm of σ_x. Remark that N is also a subterm of σ_x. Conversely, if there is an x such that N is a subterm of σ_x, then $\alpha : N$ is a subterm of σ_x. Indeed, N is clipped and σ_x is labelled, and thus $N \neq \sigma_x$ and N is a strict subterm of σ_x. In this case, residuals of $\alpha : N$ (resp. N) are at positions $\mathfrak{q}\mathfrak{p}$ (resp. $\mathfrak{q}\mathfrak{p}\mathbb{z}$) such that $\mathfrak{p}$ is the position of $\alpha : N$ in σ_x and $B|_{\mathfrak{q}} = x$.
- In other cases, neither $\alpha : N$ nor N has residuals. □

3.3.2 Redex Stability Lemma

The main result of this section is that any residual of a redex is still a redex, with same name (Redex Stability Lemma 3.10). Intermediate steps study the stability matchable forms and their matchability witnesses under reduction (Lemma 3.8), and stability of compound matching under reduction (Lemma 3.9).

Lemma 3.8. *For any matchable term X and reduction $X \to_{\mathcal{N}} X'$, X' is matchable and $|X| = |X'|$.*

Proof. By induction on the definition of matchable terms.

- If $X = [\theta]\, P \to B$, there exists P', B' such that $X' = [\theta]\, P' \to B'$, and $|X| = \varepsilon = |X'|$.
- If $X = \hat{x}$, X is irreducible, which isn't the case here.
- If X is a data structure DT, with D a labelled data structure:
  - If $X' = DT'$ with $T \to_{\mathcal{N}} T'$, then $|X| = |D| = |Y'|$.
  - If $X' = D'T$ with $D \to_{\mathcal{N}} D'$, then by induction hypothesis $|D| = |D'|$. Hence $|X| = |DT| = |D| = |D'| = |D'T| = |X'|$.
- If $X = \alpha : Z$. Since X is a matchable form, it is not a redex. Hence there exists a reduction $Z \to_{\mathcal{N}} Z'$ such that $Y' = \alpha : Z'$. Z is a matchable form. By induction hypothesis $|Z| = |Z'|$. Thus $|X| = |\alpha : Z| = \alpha|Z| = \alpha|Z'| = |\alpha : Z'| = |X'|$. □

Lemma 3.9. *Suppose $\{\!\{A/[\theta]\, P\}\!\}_p$ is defined. Then*

- *If $A \to_{\mathcal{N}} A'$, then $\{\!\{A/[\theta]\, P\}\!\}_p \to_{\mathcal{N}}^{=} \{\!\{A'/[\theta]\, P\}\!\}_p$.*
- *If $P \to_{\mathcal{N}} P'$, then $\{\!\{A/[\theta]\, P\}\!\}_p = \{\!\{A/[\theta]\, P'\}\!\}_p$.*

Where $\{\!\{A_1/[\theta]\, P_1\}\!\}_p \to_{\mathcal{N}} \{\!\{A_2/[\theta]\, P_2\}\!\}_p$ means that $\{\!\{A_i/[\theta]\, P_i\}\!\}_p = (\Gamma_i, \mu_i)$ with $\Gamma_1 = \Gamma_2$, and either $\mu_1 = \mu_2 = \bot$ or $\mu_1 = \sigma_1 \to_{\mathcal{N}} \sigma_2 = \mu_2$.

Proof. First case, by induction, with an hypothesis generalized to any terms Y and X: for any p, if $\{\!\{Y/[\theta]\, X\}\!\}_p$ is defined and $Y \to_{\mathcal{N}} Y'$ then $\{\!\{Y/[\theta]\, X\}\!\}_p \to_{\mathcal{N}}^{=} \{\!\{Y'/[\theta]\, X\}\!\}_p$. (if $p = w$, suppose Y is a matchable form).

- If $X = \alpha : Z$, then $\{\!\{Y/[\theta]\, X\}\!\}_p = \alpha : \{\!\{Y/[\theta]\, Z\}\!\}_p$ and $\{\!\{Y'/[\theta]\, X\}\!\}_p = \alpha : \{\!\{Y'/[\theta]\, Z\}\!\}_p$, with by hypothesis $\{\!\{Y/[\theta]\, Z\}\!\}_p \to^{=} \{\!\{Y'/[\theta]\, Z\}\!\}_p$.
- If $X = \hat{x}$ with $x \in \theta$, then $\{\!\{Y/[\theta]\, X\}\!\}_s = (\varepsilon, \{x \mapsto Y\}) \to (\varepsilon, \{x \mapsto Y'\}) = \{\!\{Y'/[\theta]\, X\}\!\}_s$.
- If $X = \hat{x}$ with $x \in \theta$, then $\{\!\{Y/[\theta]\, X\}\!\}_w = (|Y|, \{x \mapsto Y\}) \to_{\mathcal{N}} (|Y|, \{x \mapsto Y'\}) = (|Y'|, \{x \mapsto Y'\}) = \{\!\{Y'/[\theta]\, X\}\!\}_w$ with Lemma 3.8.
- If $Y = X = \hat{x}$ with $x \notin \theta$, Y is irreducible, which isn't the case here.
- If $X = [\tau]\, P \to B$, then $\{\!\{Y/[\theta]\, X\}\!\}_s = (\varepsilon, \bot) = \{\!\{Y'/[\theta]\, X\}\!\}_s$.
- If $X = [\tau]\, P \to B$, then $\{\!\{Y/[\theta]\, X\}\!\}_w = (|Y|, \bot) = (|Y'|, \bot) = \{\!\{Y'/[\theta]\, X\}\!\}_w$, with Lemma 3.8.
- If $X = P_1 P_2$ and $Y = A_1 A_2$, with X and Y matchable. For instance $Y' = A_1' A_2$ with $A_1 \xrightarrow{R}_{\mathcal{N}} A_1'$ (the other case is symmetrical). By induction hypothesis $\{\!\{A_1/[\theta]\, P_1\}\!\}_p \to_{\mathcal{N}}^{=} \{\!\{A_1'/[\theta]\, P_1\}\!\}_p$. Note $(\Gamma_i, \mu_i) = \{\!\{A_i/[\theta]\, P_i\}\!\}_p$ and $(\Gamma_1', \mu_1') = \{\!\{A_1'/[\theta]\, P_1\}\!\}_p$. Hypothesis adds $\Gamma_1 = \Gamma_1'$.
  - If $\mu_1 = \bot$, then $\{\!\{Y/[\theta]\, X\}\!\}_p = \{\!\{A_1/[\theta]\, P_1\}\!\}_w = (\Gamma_1, \bot) = \{\!\{A_1'/[\theta]\, P_1\}\!\}_w = \{\!\{A'/[\theta]\, P\}\!\}_p$, then $\{\!\{Y/[\theta]\, X\}\!\}_p \to_{\mathcal{N}}^{=} \{\!\{Y'/[\theta]\, X\}\!\}_p$.
  - If μ_1 is a subtitution σ_1, then $\mu_1' = \sigma_1'$.
    - If $\mu_2 = \bot$, then $\{\!\{Y/[\theta]\, X\}\!\}_p = (\Gamma_1\Gamma_2, \bot) = (\Gamma_1'\Gamma_2, \bot) = \{\!\{Y'/[\theta]\, X\}\!\}_p$.

* If μ_2 is a substitution σ_2. Equality of $\Gamma_1\Gamma_2$ and $\Gamma_1'\Gamma_2$ still holds. Moreover, by hypothesis, $\sigma_1 \to \sigma_1'$, which implies $dom(\sigma_1) = dom(\sigma_1')$. Then $\sigma_1 \uplus \sigma_2 = \bot$ if and only if $\sigma_1' \uplus \sigma_2 = \bot$. Else $\sigma_1 \uplus \sigma_2 \to_{\mathcal{N}} \sigma_1' \uplus \sigma_2$. Finally $\{\!\{Y/[\theta]\,X\}\!\}_p = (\Gamma_1\Gamma_2, \sigma_1 \uplus \sigma_2) \to_{\mathcal{N}}^{=} (\Gamma_1'\Gamma_2, \sigma_1' \uplus \sigma_2) = \{\!\{Y/[\theta]\,X\}\!\}_p$.

- If X is matchable with $Y = \alpha : Z$ and $Y' = \alpha : Z'$, then $\{\!\{Y/[\theta]\,X\}\!\}_p = \alpha : \{\!\{Z/[\theta]\,X\}\!\}_p$ and $\{\!\{Y'/[\theta]\,X\}\!\}_p = \alpha : \{\!\{Z'/[\theta]\,X\}\!\}_p$ with by induction hypothesis $\{\!\{Z/[\theta]\,X\}\!\}_p \to_{\mathcal{N}}^{=} \{\!\{Z'/[\theta]\,X\}\!\}_p$ (Y can't be a redex, for otherwise $\{\!\{Y/[\theta]\,X\}\!\}_p$ wouln't be defined).

- If X and Y are other matchable terms, then $\{\!\{Y/[\theta]\,X\}\!\}_p = (|Y||X|, \bot) = (|Y'||X|) = \{\!\{Y'/[\theta]\,X\}\!\}_p$, with Lemma 3.8.

- Otherwise, $\{\!\{Y/[\theta]\,X\}\!\}_p$ is `wait`, which isn't the case here.

Finally, since $\{\!\{Y/[\theta]\,X\}\!\}_p \to_{\mathcal{N}}^{=} \{\!\{Y'/[\theta]\,X\}\!\}_p$, $\{Y/[\theta]\,X\} \to_{\mathcal{N}}^{=} \{Y'/[\theta]\,X\}$ also holds, because if $\{\!\{Y/[\theta]\,X\}\!\}_p$ is a substitution, then $\{\!\{Y'/[\theta]\,X\}\!\}_p$ is one too, with same domain (and conversely). The second point is symmetrical. □

Lemma 3.10 (Redex Stability Lemma)**.** *For any term X with a redex R_a of name Ω_a. If $X \xrightarrow{R} X'$ and R_r is a residual of R_a in X', then R_r is still a redex of name Ω_a.*

Proof. By case on the residual relation. Note $R_a = \alpha_a : ((\Gamma_a \cdot [\theta_a]\,P_a \to B_a)A_a)$, with $\{A_a/[\theta_a]\,P_a\} = (\Delta_a, \mu_a)$. The name of R_a is $\Omega_a = \alpha_a\Gamma_a\Delta_a$. Note $R = \alpha : ((\Gamma \cdot [\theta]\,P \to B)A)$, with $\{A/[\theta]\,P\} = (\Delta, \mu)$. The name of R is $\Omega = \alpha\Gamma\Delta$.

- If R_a is disjoint from R, $R_r = R_a$, and position is still active.

- If R is a strict subterm of R_a, one of the following holds: $B_a \xrightarrow{R} B_a'$, $A_a \xrightarrow{R} A_a'$ or $P_a \xrightarrow{R} P_a'$. In the former case, $R_r = \alpha_a : ((\Gamma_a \cdot [\theta_a]\,P_a \to B_a')A_a)$ and the name is still $\Omega_a = \alpha_a\Gamma_a\Delta_a$. In the case $A_a \xrightarrow{R} A_a'$, $R_r = \alpha_a : ((\Gamma_a \cdot [\theta_a]\,P_a \to B_a)A_a')$, and by Lemma 3.9 $\{A_a/[\theta_a]\,P_a\} \to^{=} \{A_a'/[\theta]\,P_a\}$. In particular $\{A_a'/[\theta_a]\,P_a\} = (\Delta_a, \mu_a')$ and thus the name of R_r is still $\Omega_a = \alpha_a\Gamma_a\Delta_a$. Third case is similar.

- If R_a is a subterm of B and μ is a substitution σ, then $R_r = (\Omega(\!|\theta|\!)R_a)^\sigma$. Suppose $fv(R_a) \cap dom(\sigma) \neq \emptyset$. Since σ is a substitution, $dom(\sigma) = \theta$. Thus $fv(R_a) \cap \theta \neq \emptyset$ and R_a was not a redex in $[\theta]\,P \to B$. Then suppose $fv(R_a) \# dom(\sigma)$: $R_r = R_a$ and the name is the same.

- If μ is a substitution and R_a is a subterm of A which appears in the codomain of μ. Then $R_r = R_a$. □

3.3.3 Confluence Theorem

Theorem 3.11. WPPC *is confluent.*

Proof. Particular case of Confluence Theorem 3.12 for *LWPPC*. □

Theorem 3.12 (Confluence Theorem)**.** LWPPC *is confluent.*

Proof. Particular case of Parametric Confluence Theorem 3.21. □

The following constructions lead to the proof of confluence of $\to_{\mathcal{N}}$ for any set of names $\mathcal{N}$ (Parametric Confluence Theorem 3.21), using Tait and Martin-Löfs technique. It is based on a notion of parallel reduction $\Rightarrow_{\mathcal{N}}$ which satisfies the one-step diamond property or *strong confluence* (Diamond Lemma 3.19) and the inclusions $\to_{\mathcal{N}} \subseteq \Rightarrow_{\mathcal{N}} \subseteq \to_{\mathcal{N}}^*$ (Simulation Lemma 3.20). A

key step is the establishment of results of stability similar to those presented in Section 3.3.2: stability of matchable forms and their matchability witnesses under parallel reduction (Lemma 3.13), stability of compound matching under parallel reduction (Lemma 3.14) and stability of redexes under parallel reduction (Corollary 3.15). Other significant results are commutations between relabelling, substitution, and parallel reduction (Lemmas 3.17 and 3.18).

In this section, notation $\overrightarrow{R}$ is used for a list of redexes.

In order to clarify the proof, the two following cases of the reduction rule are separated.

$$\begin{array}{rcrcl} \mu = \bot & : & R & \xrightarrow{R} & \ulcorner \Omega \urcorner : \bot \\ \mu = \sigma & : & R & \xrightarrow{R} & \ulcorner \Omega \urcorner : (\Omega \langle\!| \theta |\!\rangle B)^{\sigma} \end{array}$$

Parallel reduction is defined by the following rule.

$$X \overset{\emptyset}{\Rightarrow}_{\mathcal{N}} X \qquad \frac{X \overset{\overrightarrow{R}}{\Rightarrow}_{\mathcal{N}} X'}{X \Rightarrow_{\mathcal{N}} X'} \qquad \frac{X \overset{\overrightarrow{R}}{\Rightarrow}_{\mathcal{N}} X'}{\alpha : X \overset{\overrightarrow{R}}{\Rightarrow}_{\mathcal{N}} \alpha : X'}$$

$$\frac{T_1 \overset{\overrightarrow{R_1}}{\Rightarrow}_{\mathcal{N}} T_1' \qquad T_2 \overset{\overrightarrow{R_2}}{\Rightarrow}_{\mathcal{N}} T_2'}{T_1 T_2 \overset{\overrightarrow{R_1}\overrightarrow{R_2}}{\Rightarrow}_{\mathcal{N}} T_1' T_2'} \qquad \frac{P \overset{\overrightarrow{R_P}}{\Rightarrow}_{\mathcal{N}} P' \qquad B \overset{\overrightarrow{R_B}}{\Rightarrow}_{\mathcal{N}} B' \qquad \theta \# \overrightarrow{R_B}}{[\theta]\, P \to B \overset{\overrightarrow{R_P}\overrightarrow{R_B}}{\Rightarrow}_{\mathcal{N}} [\theta]\, P' \to B'}$$

$$\texttt{fail}\ \frac{\begin{array}{c} \theta \# \overrightarrow{R_B} \qquad P \overset{\overrightarrow{R_P}}{\Rightarrow}_{\mathcal{N}} P' \qquad \{A/[\theta]\, P\} = (\Delta, \bot) \\ B \overset{\overrightarrow{R_B}}{\Rightarrow}_{\mathcal{N}} B' \qquad A \overset{\overrightarrow{R_A}}{\Rightarrow}_{\mathcal{N}} A' \qquad \{A'/[\theta]\, P'\} = (\Delta', \bot) \end{array}}{R = \alpha : ((\Gamma \cdot [\theta]\, P \to B)A) \overset{R\overrightarrow{R_P}\overrightarrow{R_B}\overrightarrow{R_A}}{\Rightarrow}_{\mathcal{N}} \ulcorner \alpha \Gamma \Delta' \urcorner : \bot}\ \alpha\Gamma\Delta' \in \mathcal{N}$$

$$\texttt{substitution}\ \frac{\begin{array}{c} \theta \# \overrightarrow{R_B} \qquad P \overset{\overrightarrow{R_P}}{\Rightarrow}_{\mathcal{N}} P' \qquad \{A/[\theta]\, P\} = (\Delta, \sigma) \\ B \overset{\overrightarrow{R_B}}{\Rightarrow}_{\mathcal{N}} B' \qquad A \overset{\overrightarrow{R_A}}{\Rightarrow}_{\mathcal{N}} A' \qquad \{A'/[\theta]\, P'\} = (\Delta', \sigma') \end{array}}{R = \alpha : ((\Gamma \cdot [\theta]\, P \to B)A) \overset{R\overrightarrow{R_P}\overrightarrow{R_B}\overrightarrow{R_A}}{\Rightarrow}_{\mathcal{N}} \ulcorner \alpha \Gamma \Delta' \urcorner : (\alpha\Gamma\Delta' \langle\!| \theta |\!\rangle B')^{\sigma'}}\ \alpha\Gamma\Delta' \in \mathcal{N}$$

For σ and σ' two substitutions with same domain $\theta = x_1 ... x_n$, write $\sigma \overset{\overrightarrow{R_1} ... \overrightarrow{R_n}}{\Rightarrow}_{\mathcal{N}} \sigma'$ when $\sigma_{x_i} \overset{\overrightarrow{R_i}}{\Rightarrow}_{\mathcal{N}} \sigma'_{x_i}$ for all $i \in \{1...n\}$.

Lemma 3.13 (Lemma 3.8 parallel)**.** *If M is a matchable form (resp. data structure) and $M \Rightarrow_{\mathcal{N}} M'$, then M' is a matchable form (resp. data structure) and $|M'| = |M|$.*

Proof. By induction on $M \Rightarrow_{\mathcal{N}} M'$.

- If $M \overset{\emptyset}{\Rightarrow}_{\mathcal{N}} M = M'$, the result is obvious.
- If $M = \alpha : X \Rightarrow_{\mathcal{N}} \alpha : X' = M'$ with $X \Rightarrow_{\mathcal{N}} X'$. By induction hypothesis, if X is a data structure then X' is a data structure, else X and X' are matchable forms. In both case the equality $|X| = |X'|$ holds. Thus $|M| = |\alpha : X| = \alpha|X| = \alpha|X'| = |\alpha : X'| = |M'|$.
- If $M = DT \Rightarrow_{\mathcal{N}} D'T' = M'$ with $D \Rightarrow_{\mathcal{N}} D'$, $T \Rightarrow_{\mathcal{N}} T'$ and D is a data structure. Then by induction hypothesis D' is a data structure with $|D'| = |D|$. Thus M' is also a data structure, and $|M| = |DT| = |D| = |D'| = |D'T'| = |M'|$.

- If $M = [\theta]\, P \to B \Rightarrow_{\mathcal{N}} [\theta]\, P' \to B' = M'$, then M' is a matchable form and $|M'| = \varepsilon = |M|$.
- The other cases of parallel reduction do not concern matchable forms. □

Lemma 3.14 (Lemma 3.9 parallel)**.** *If $\{A/[\theta]\, P\}$ is defined, and $P \Rightarrow_{\mathcal{N}} P'$ and $A \Rightarrow_{\mathcal{N}} A'$, then $\{A/[\theta]\, P\} \Rightarrow_{\mathcal{N}} \{A'/[\theta]\, P'\}$.*

Proof. For any Y, X such that $\{\!\{Y/[\theta]\, X\}\!\}_p$ is defined, with $X \Rightarrow_{\mathcal{N}} X'$ and $Y \Rightarrow_{\mathcal{N}} Y'$, $\{\!\{Y/[\theta]\, X\}\!\}_p \Rightarrow_{\mathcal{N}} \{\!\{Y'/[\theta]\, X'\}\!\}_p$, by induction on the definition of $\{\!\{Y/[\theta]\, X\}\!\}_p$. (if $p = w$, suppose Y is a matchable form)

- If $X = \alpha : Z$, then $\{\!\{Y/[\theta]\, X\}\!\}_p = \alpha : \{\!\{Y/[\theta]\, Z\}\!\}_p$ and $\{\!\{Y'/[\theta]\, X'\}\!\}_p = \alpha : \{\!\{Y'/[\theta]\, Z'\}\!\}_p$, with by hypothesis $\{\!\{Y/[\theta]\, Z\}\!\}_p \Rightarrow_{\mathcal{N}} \{\!\{Y'/[\theta]\, Z'\}\!\}_p$ (X can't be a redex, for otherwise $\{\!\{Y/[\theta]\, X\}\!\}_p$ wouln't be defined).
- If $X = \hat{x}$ with $x \in \theta$. First remark that X is a normal form, and thus $X' = X$. Then $\{\!\{Y/[\theta]\, X\}\!\}_s = (\varepsilon, \{x \mapsto Y\}) \Rightarrow_{\mathcal{N}} (\varepsilon, \{x \mapsto Y'\}) = \{\!\{Y'/[\theta]\, X\}\!\}_s = \{\!\{Y'/[\theta]\, X'\}\!\}_s$.
- If $X = \hat{x}$ with $x \in \theta$, then $X' = X$ and $\{\!\{Y/[\theta]\, X\}\!\}_w = (|Y|, \{x \mapsto Y\}) \Rightarrow_{\mathcal{N}} (|Y|, \{x \mapsto Y'\}) = (|Y'|, \{x \mapsto Y'\}) = \{\!\{Y'/[\theta]\, X\}\!\}_w = \{\!\{Y'/[\theta]\, X'\}\!\}_w$ with Lemma 3.13.
- If $Y = X = \hat{x}$ with $x \notin \theta$, then $Y' = Y$, $X' = X$ and $\{\!\{Y/[\theta]\, X\}\!\}_p \stackrel{\emptyset}{\Rightarrow}_{\mathcal{N}} \{\!\{Y'/[\theta]\, X'\}\!\}_p$.
- If $X = [\tau]\, P \to B$, then there are P' and B' such that $X' = [\tau]\, P' \to B'$ and $\{\!\{Y/[\theta]\, X\}\!\}_s = (\varepsilon, \bot) = \{\!\{Y'/[\theta]\, X'\}\!\}_s$.
- If $X = [\tau]\, P \to B$, then there are P' and B' such that $X' = [\tau]\, P' \to B'$ and $\{\!\{Y/[\theta]\, X\}\!\}_w = (|Y|, \bot) = (|Y'|, \bot) = \{\!\{Y'/[\theta]\, X'\}\!\}_w$, by Lemma 3.13.
- If $X = P_1 P_2$ and $Y = A_1 A_2$, with X and Y matchable. $Y' = A'_1 A'_2$ with $A_1 \Rightarrow_{\mathcal{N}} A'_1$, $A_2 \Rightarrow_{\mathcal{N}} A'_2$ and $X' = P'_1 P'_2$ with $P_1 \Rightarrow_{\mathcal{N}} P'_1$, $P_2 \Rightarrow_{\mathcal{N}} P'_2$. By induction hypothesis $(\Gamma_1, \mu_1) = \{\!\{A_1/[\theta]\, P_1\}\!\}_w \Rightarrow_{\mathcal{N}} \{\!\{A'_1/[\theta]\, P'_1\}\!\}_w = (\Gamma_1, \mu'_1)$ and $(\Gamma_2, \mu_2) = \{\!\{A_2/[\theta]\, P_2\}\!\}_s \Rightarrow_{\mathcal{N}} \{\!\{A'_2/[\theta]\, P'_2\}\!\}_s = (\Gamma_2, \mu'_2)$.
  - If $\mu_1 = \bot$, then $\mu'_1 = \bot$ and $\{\!\{Y/[\theta]\, X\}\!\}_p = \{\!\{A_1/[\theta]\, P_1\}\!\}_w = (\Gamma_1, \bot) = \{\!\{A'_1/[\theta]\, P'_1\}\!\}_w = \{\!\{Y'/[\theta]\, X'\}\!\}_p$, then $\{\!\{Y/[\theta]\, X\}\!\}_p \stackrel{\emptyset}{\Rightarrow}_{\mathcal{N}} \{\!\{Y'/[\theta]\, X'\}\!\}_p$.
  - If μ_1 is a subtitution σ_1, then $\mu'_1 = \sigma'_1$.
    - If $\mu_2 = \bot$, then $\mu'_2 = \bot$ and $\{\!\{Y/[\theta]\, X\}\!\}_p = (\Gamma_1\Gamma_2, \bot) = \{\!\{Y'/[\theta]\, X'\}\!\}_p$.
    - If μ_2 is a substitution σ_2. Then $\mu'_2 = \sigma'_2$. Moreover, by hypothesis, $\sigma_1 \Rightarrow_{\mathcal{N}} \sigma'_1$ and $\sigma_2 \Rightarrow_{\mathcal{N}} \sigma'_2$, which implies $dom(\sigma_1) = dom(\sigma'_1)$ and $dom(\sigma_2) = dom(\sigma'_2)$. Then $\sigma_1 \uplus \sigma_2 = \bot$ if and only if $\sigma'_1 \uplus \sigma'_2 = \bot$. Else $\sigma_1 \uplus \sigma_2 \Rightarrow_{\mathcal{N}} \sigma'_1 \uplus \sigma'_2$. Finally $\{\!\{Y/[\theta]\, X\}\!\}_p = (\Gamma_1\Gamma_2, \sigma_1 \uplus \sigma_2) \Rightarrow_{\mathcal{N}} (\Gamma_1\Gamma_2, \sigma'_1 \uplus \sigma'_2) = \{\!\{Y'/[\theta]\, X'\}\!\}_p$.
- If X is matchable with $Y = \alpha : Z$ and $Y' = \alpha : Z'$, then $\{\!\{Y/[\theta]\, X\}\!\}_p = \alpha : \{\!\{Z'/[\theta]\, X'\}\!\}_p$ and $\{\!\{Y'/[\theta]\, X'\}\!\}_p = \alpha : \{\!\{Z'/[\theta]\, X'\}\!\}_p$ with by induction hypothesis $\{\!\{Z/[\theta]\, X\}\!\}_p \Rightarrow_{\mathcal{N}} \{\!\{Z'/[\theta]\, X'\}\!\}_p$ (Y can't be a redex, for otherwise $\{\!\{Y/[\theta]\, X\}\!\}_p$ wouln't be defined).
- If X and Y are other matchable terms, then $\{\!\{Y/[\theta]\, X\}\!\}_p = (|Y||X|, \bot) = (|Y'||X'|) = \{\!\{Y'/[\theta]\, X'\}\!\}_p$, with Lemma 3.8.
- Otherwise, $\{\!\{Y/[\theta]\, X\}\!\}_p$ is `wait`, which isn't the case here.

Finally, since $\{\!\{Y/[\theta]\,X\}\!\}_p \Rightarrow_{\mathcal{N}} \{\!\{Y'/[\theta]\,X'\}\!\}_p$, $\{Y/[\theta]\,X\} \Rightarrow_{\mathcal{N}} \{Y'/[\theta]\,X'\}$ also holds, because if $\{\!\{Y/[\theta]\,X\}\!\}_p$ is a substitution, then $\{\!\{Y'/[\theta]\,X'\}\!\}_p$ is one too, with same domain (and conversely). □

Corollary 3.15. *Rules* `fail` *and* `substitution` *can be replaced respectively by the following:*

$$\texttt{fail\_3.15}\ \frac{\begin{array}{ccc}\theta\#\overrightarrow{R_B} & P \overset{\overrightarrow{R_P}}{\Longrightarrow}_{\mathcal{N}} P' & \\ B \overset{\overrightarrow{R_B}}{\Longrightarrow}_{\mathcal{N}} B' & A \overset{\overrightarrow{R_A}}{\Longrightarrow}_{\mathcal{N}} A' & \{A/[\theta]\,P\} = (\Delta, \bot)\end{array}}{R = \alpha : ((\Gamma \cdot [\theta]\,P \to B)A) \overset{R\overrightarrow{R_P}\overrightarrow{R_B}\overrightarrow{R_A}}{\Longrightarrow}_{\mathcal{N}} \ulcorner \alpha\Gamma\Delta \urcorner : \bot}\ \alpha\Gamma\Delta \in \mathcal{N}$$

$$\texttt{substitution\_3.15}\ \frac{\begin{array}{ccc}\theta\#\overrightarrow{R_B} & P \overset{\overrightarrow{R_P}}{\Longrightarrow}_{\mathcal{N}} P' & \{A/[\theta]\,P\} = (\Delta, \sigma) \\ B \overset{\overrightarrow{R_B}}{\Longrightarrow}_{\mathcal{N}} B' & A \overset{\overrightarrow{R_A}}{\Longrightarrow}_{\mathcal{N}} A' & \{A'/[\theta]\,P'\} = (\Delta, \sigma')\end{array}}{R = \alpha : ((\Gamma \cdot [\theta]\,P \to B)A) \overset{R\overrightarrow{R_P}\overrightarrow{R_B}\overrightarrow{R_A}}{\Longrightarrow}_{\mathcal{N}} \ulcorner \alpha\Gamma\Delta \urcorner : (\alpha\Gamma\Delta(\!|\theta|\!)B')^{\sigma'}}\ \alpha\Gamma\Delta \in \mathcal{N}$$

Lemma 3.16. *If $X \Rightarrow_{\mathcal{N}} X'$, then $fv(X') \subseteq fv(X)$.*

Proof. Prove first that if $\{\!\{Y/[\theta]\,Z\}\!\}_p = (\Delta, \sigma)$ then $fv(\sigma) \subseteq fv(Y)$, by induction on $\{\!\{Y/[\theta]\,Z\}\!\}_p$. Then conclude by induction on $X \Rightarrow_{\mathcal{N}} X'$ (for the `fail` rule, remember that $\bot$ is a closed term). □

Lemma 3.17. *If $X \overset{\overrightarrow{R}}{\Longrightarrow}_{\mathcal{N}} X'$ and $\theta\#\overrightarrow{R}$, then there is $\overrightarrow{R^*}$ with $fv(\overrightarrow{R}) = fv(\overrightarrow{R^*})$ such that $\Omega(\!|\theta|\!)X \overset{\overrightarrow{R^*}}{\Longrightarrow}_{\mathcal{N}} \Omega(\!|\theta|\!)X'$.*

Proof. Induction on $X \overset{\overrightarrow{R}}{\Longrightarrow}_{\mathcal{N}} X'$.

First remark that if $\theta\#X$, by Lemma 3.16, $\theta\#X'$, and $\Omega(\!|\theta|\!)X = X \overset{\overrightarrow{R}}{\Longrightarrow}_{\mathcal{N}} X' = \Omega(\!|\theta|\!)X'$. Now suppose $\theta \cap X \neq \emptyset$.

- If $X \overset{\emptyset}{\Rightarrow}_{\mathcal{N}} X'$, then $X' = X$, $\Omega(\!|\theta|\!)X = \Omega(\!|\theta|\!)X'$ and $\Omega(\!|\theta|\!)X \overset{\emptyset}{\Rightarrow}_{\mathcal{N}} \Omega(\!|\theta|\!)X'$.
- If $X = \alpha : Z \overset{\overrightarrow{R}}{\Rightarrow}_{\mathcal{N}} \alpha : Z' = X'$. By induction hypothesis $\Omega(\!|\theta|\!)Z \overset{\overrightarrow{R^*}}{\Longrightarrow}_{\mathcal{N}} \Omega(\!|\theta|\!)Z'$ with $fv(\overrightarrow{R^*}) = fv(\overrightarrow{R})$. Thus $\Omega(\!|\theta|\!)\alpha : Z = [\Omega, \alpha] : (\Omega(\!|\theta|\!)Z) \overset{\overrightarrow{R^*}}{\Longrightarrow}_{\mathcal{N}} [\Omega, \alpha] : (\Omega(\!|\theta|\!)Z' = \Omega(\!|\theta|\!)\alpha : Z'$.
- If $X = T_1T_2 \overset{\overrightarrow{R_1}\overrightarrow{R_2}}{\Longrightarrow}_{\mathcal{N}} T_1'T_2' = X'$ with $T_1 \overset{\overrightarrow{R_1}}{\Longrightarrow}_{\mathcal{N}} T_1'$ and $T_2 \overset{\overrightarrow{R_2}}{\Longrightarrow}_{\mathcal{N}} T_2'$. By induction hypothesis there are $\overrightarrow{R_1^*}$ and $\overrightarrow{R_2^*}$ with $fv(\overrightarrow{R_1}) = fv(\overrightarrow{R_1^*})$ and $fv(\overrightarrow{R_2}) = fv(\overrightarrow{R_2^*})$ such that $\Omega(\!|\theta|\!)T_1 \overset{\overrightarrow{R_1^*}}{\Longrightarrow}_{\mathcal{N}} \Omega(\!|\theta|\!)T_1'$ and $\Omega(\!|\theta|\!)T_2 \overset{\overrightarrow{R_2^*}}{\Longrightarrow}_{\mathcal{N}} \Omega(\!|\theta|\!)T_2'$. Thus $\Omega(\!|\theta|\!)T_1T_2 = (\Omega(\!|\theta|\!)T_1)(\Omega(\!|\theta|\!)T_2) \overset{\overrightarrow{R_1^*}\overrightarrow{R_2^*}}{\Longrightarrow}_{\mathcal{N}} (\Omega(\!|\theta|\!)T_1')(\Omega(\!|\theta|\!)T_2') = \Omega(\!|\theta|\!)T_1'T_2'$. With $fv(\overrightarrow{R_1^*}\overrightarrow{R_2^*}) = fv(\overrightarrow{R_1}\overrightarrow{R_2})$.
- If $X = [\tau]\,P \to B$, as in the previous case $[\tau]\,P \to B \overset{\overrightarrow{R_P}\overrightarrow{R_B}}{\Longrightarrow}_{\mathcal{N}} [\tau]\,P' \to B'$, with in addition $\tau\#\overrightarrow{R_B}$. By induction hypothesis, in particular $\Omega(\!|\theta|\!)B \overset{\overrightarrow{R_B^*}}{\Longrightarrow}_{\mathcal{N}} \Omega(\!|\theta|\!)B'$ with $fv(\overrightarrow{R_B^*}) = fv(\overrightarrow{R_B})$. Hence $\tau\#\overrightarrow{R_B^*}$, which allows to derive the reduction $[\tau]\,P \to B \overset{\overrightarrow{R_P^*}\overrightarrow{R_B^*}}{\Longrightarrow}_{\mathcal{N}} [\tau]\,P' \to B'$ (with $fv(\overrightarrow{R_P^*}\overrightarrow{R_B^*}) = fv(\overrightarrow{R_P}\overrightarrow{R_B})$).

- If reduction is by the `fail` or the `substitution` rule, then X is in $\overrightarrow{R}$. But $\theta \# R$, and thus $\theta \# X$, which is not the case. □

Lemma 3.18. *If $T \overset{\overrightarrow{R}}{\Rightarrow}_{\mathcal{N}} T'$, $\sigma \Rightarrow_{\mathcal{N}} \sigma'$ and $dom(\sigma) \# \overrightarrow{R}$, then $T^\sigma \Rightarrow_{\mathcal{N}} T'^{\sigma'}$.*

Proof. First remark that σ and σ' have the same domain, by definition of $\sigma \Rightarrow_{\mathcal{N}} \sigma'$. Write $\tau = dom(\sigma) = dom(\sigma')$.

The following strengthened hypothesis is proved: for any $X \overset{\overrightarrow{R}}{\Rightarrow}_{\mathcal{N}} X'$, $\sigma \Rightarrow_{\mathcal{N}} \sigma'$ with $dom(\sigma) \# \overrightarrow{R}$, there exists $\overrightarrow{R^*}$ with $fv(\overrightarrow{R^*}) \subseteq fv(\overrightarrow{R}) \cup fv(\sigma)$ such that $X^\sigma \overset{\overrightarrow{R^*}}{\Longrightarrow} X'^{\sigma'}$. By induction on $X \overset{\overrightarrow{R}}{\Rightarrow}_{\mathcal{N}} X'$.

- If $X \overset{\emptyset}{\Rightarrow}_{\mathcal{N}} X = X'$, then conclude by induction on X:
  - If $X = x$ with $x \in \tau$, then $X^\sigma = \sigma_x \Rightarrow_{\mathcal{N}} \sigma'_x = X^{\sigma'}$.
  - If $X = x$ with $x \notin \tau$, then $X^\sigma = x = X^{\sigma'}$.
  - If $X = \hat{x}$, then $X^\sigma = \hat{x} = X^{\sigma'}$.
  - If $X = T_1 T_2$, then

$$\begin{aligned} X^\sigma &= (T_1 T_2)^\sigma \\ &= T_1^\sigma T_2^\sigma \end{aligned}$$

    By induction hypothesis (on X), $T_1^\sigma \Rightarrow_{\mathcal{N}} T_1^{\sigma'}$ et $T_2^\sigma \Rightarrow_{\mathcal{N}} T_2^{\sigma'}$. Thus:

$$\begin{aligned} X^\sigma &\Rightarrow_{\mathcal{N}} T_1^{\sigma'} T_2^{\sigma'} \\ &= (T_1 T_2)^{\sigma'} \\ &= X^{\sigma'} \end{aligned}$$

  - If $X = [\theta]\, P \to B$, then $X^\sigma = [\theta]\, P^\sigma \to B^\sigma$ with $\theta \# \sigma$. By induction hypothesis (on X), $P^\sigma \Rightarrow_{\mathcal{N}} P^{\sigma'}$ and $B^\sigma \overset{\overrightarrow{R_B}}{\Longrightarrow}_{\mathcal{N}} B^{\sigma'}$. But $fv(\overrightarrow{R_B}) \subseteq fv(\sigma)$, and $\theta \# \sigma$, thus $\theta \# \overrightarrow{R_B}$ and $[\theta]\, P^\sigma \to B^\sigma \Rightarrow_{\mathcal{N}} [\theta]\, P^{\sigma'} \to B^{\sigma'}$. Hence $X^\sigma \Rightarrow_{\mathcal{N}} X^{\sigma'}$.
  - If $X = \alpha : Z$, conclusion is straightforward with induction hypothesis (on X).
- If $X = T_1 T_2 \Rightarrow_{\mathcal{N}} T'_1 T'_2 = X'$, with $T_1 \Rightarrow_{\mathcal{N}} T'_1$ and $T_2 \Rightarrow_{\mathcal{N}} T'_2$. By induction hypothesis, $T_1^\sigma \Rightarrow_{\mathcal{N}} (T'_1)^{\sigma'}$ and $T_2^\sigma \Rightarrow_{\mathcal{N}} (T'_2)^{\sigma'}$. Then $X^\sigma \Rightarrow_{\mathcal{N}} X'^{\sigma'}$.
- If $X = [\theta]\, P \to B \Rightarrow_{\mathcal{N}} [\theta]\, P' \to B'$, with $P \Rightarrow_{\mathcal{N}} P'$, $B \overset{\overrightarrow{R_B}}{\Longrightarrow}_{\mathcal{N}} B'$ and $\theta \# \overrightarrow{R_B}$. Then $X^\sigma = [\theta]\, P^\sigma \to B^\sigma$ with $\theta \# \sigma$. By induction hypothesis, $P^\sigma \Rightarrow_{\mathcal{N}} (P')^{\sigma'}$ and $B^\sigma \overset{\overrightarrow{R_B^*}}{\Longrightarrow}_{\mathcal{N}} B'^{\sigma'}$, where $fv(\overrightarrow{R_B^*}) \subseteq fv(\overrightarrow{R_B}) \cup \sigma$. Then $\theta \# \overrightarrow{R_B^*}$ and $X^\sigma \Rightarrow X'^{\sigma'}$.
- If $X = \alpha : Z \Rightarrow_{\mathcal{N}} \alpha : Z' = X'$ with $Z \Rightarrow_{\mathcal{N}} Z'$, conclusion is straightforward with induction hypothesis.
- If reduction is by the `fail` or the `substitution` rule, then in particular X itself is in $\overrightarrow{R}$. Since $\tau \# \overrightarrow{R}$, then $\tau \# X$ and $X^\sigma = X$. Moreover, $fv(X') \subseteq fv(X)$ by Lemma 3.16, and thus $\tau \# X'$ and $(X')^{\sigma'} = X'$. Finally $X^\sigma = X \overset{\overrightarrow{R}}{\Rightarrow}_{\mathcal{N}} X' = (X')^{\sigma'}$. □

Lemma 3.19 (Diamond). *Relation $\Rightarrow_{\mathcal{N}}$ has the diamond property : if $X_l \,{}_{\mathcal{N}}\!\Leftarrow X \Rightarrow_{\mathcal{N}} X_r$, then there exists X_c such that $X_l \Rightarrow_{\mathcal{N}} X_c \,{}_{\mathcal{N}}\!\Leftarrow X_r$. (l, r and c stand for **l**eft path, **r**ight path and **c**onfluence)*

Proof. Strengthened hypothesis : if $X_l \; \mathcal{N} \overleftarrow{\overrightarrow{R_l}} X \overrightarrow{\overrightarrow{R_r}}_{\mathcal{N}} X_r$, then there exists X_c, R_l^*, R_r^* such that $X_l \overrightarrow{\overrightarrow{R_l^*}}_{\mathcal{N}} X_c \; \mathcal{N} \overleftarrow{\overrightarrow{R_r^*}} X_r$, with $fv(\overrightarrow{R_l^*}) \subseteq fv(\overrightarrow{R_l}) \cup fv(\overrightarrow{R_r})$ and $fv(\overrightarrow{R_r^*}) \subseteq fv(\overrightarrow{R_l}) \cup fv(\overrightarrow{R_r})$.
By induction on $X \Rightarrow_{\mathcal{N}} X_l$.

- If $X \overrightarrow{\emptyset}_{\mathcal{N}} X = X_l$, conclusion is straightforward.
- If $X = \alpha : Z \overrightarrow{\overrightarrow{R_l}}_{\mathcal{N}} \alpha : Z_l = X_l$ with $Z \overrightarrow{\overrightarrow{R_l}}_{\mathcal{N}} Z_l$, then there are two cases on $X \Rightarrow_{\mathcal{N}} X_r$:
  - If $X = \alpha : Z \overrightarrow{\overrightarrow{R_r}}_{\mathcal{N}} \alpha : Z_r = X_r$ with $Z \overrightarrow{\overrightarrow{R_r}}_{\mathcal{N}} Z_r$, then by induction hypothesis $Z_l \overrightarrow{\overrightarrow{R_l^*}}_{\mathcal{N}} Z_c \; \mathcal{N} \overleftarrow{\overrightarrow{R_r^*}} Z_r$, and $X_l \overrightarrow{\overrightarrow{R_l^*}}_{\mathcal{N}} \alpha : Z_c \; \mathcal{N} \overleftarrow{\overrightarrow{R_r^*}} X_r$, with asked conditions on $\overrightarrow{R_l^*}$ and $\overrightarrow{R_r^*}$.
  - If $X \overrightarrow{\overrightarrow{R_r}}_{\mathcal{N}} X_r$ with the `fail_3.15` rule or the `substitution_3.15` rule, then $X = \alpha : ((\Gamma \cdot [\theta]\, P \to B)A)$, with $\{A/[\theta]\, P\} = (\Delta, \mu)$, $P \Rightarrow_{\mathcal{N}} P_r$, $A \Rightarrow_{\mathcal{N}} A_r$, $B \overrightarrow{\overrightarrow{R_{Br}}}_{\mathcal{N}} B_r$, $\theta \# \overrightarrow{R_{Br}}$, $\{A_r/[\theta]\, P_r\} = (\Delta, \mu_r)$, and $\alpha\Gamma\Delta \in \mathcal{N}$.
    With such a form for X, reduction $X \Rightarrow_{\mathcal{N}} X_l$ is derived from $P \Rightarrow_{\mathcal{N}} P_l$, $A \Rightarrow_{\mathcal{N}} A_l$ and $B \overrightarrow{\overrightarrow{R_{Bl}}}_{\mathcal{N}} B_l$ with $\theta \# \overrightarrow{R_{Bl}}$.
    If $X \Rightarrow_{\mathcal{N}} X_r$ by the `fail_3.15` rule, then $X_r = \ulcorner \alpha\Gamma\Delta \urcorner : \bot$. Since $\mu = \bot$ and $\alpha\Gamma\Delta \in \mathcal{N}$, by `fail_3.15` $X_l \overrightarrow{X_l}_{\mathcal{N}} \ulcorner \alpha\Gamma\Delta \urcorner : \bot$. Hence $X_l \overrightarrow{X_l}_{\mathcal{N}} X_r$. Since $X_r \overrightarrow{\emptyset}_{\mathcal{N}} X_r$ the diamond is completed. Suppose reduction $X \Rightarrow_{\mathcal{N}} X_r$ is by the `substitution_3.15` rule, then μ_r is a substitution σ_r and $X_r = \ulcorner \alpha\Gamma\Delta \urcorner : (\alpha\Gamma\Delta(\!|\theta|\!)B_r)^{\sigma_r}$.
    By induction hypothesis, $P_l \Rightarrow_{\mathcal{N}} P_c \; \mathcal{N} \Leftarrow P_r$, $A_l \Rightarrow_{\mathcal{N}} A_c \; \mathcal{N} \Leftarrow A_r$, and $B_l \overrightarrow{\overrightarrow{R_{Bl}^*}}_{\mathcal{N}} B_c \; \mathcal{N} \overleftarrow{\overrightarrow{R_{Br}^*}} B_r$ with $fv(\overrightarrow{R_{Bl}^*}) \subseteq fv(\overrightarrow{R_{Bl}}) \cup fv(\overrightarrow{R_{Br}})$. Since $\theta \# \overrightarrow{R_{bl}}$ and $\theta \# \overrightarrow{R_{Br}}$, in particular $\theta \# \overrightarrow{R_{Bl}^*}$, and similarly $\theta \# \overrightarrow{R_{Br}^*}$.
    Finally $\{A_c/[\theta]\, P_c\} = (\Delta, \sigma_c)$ and the `substitution_3.15` applies: $X_l \Rightarrow_{\mathcal{N}} \ulcorner \alpha\Gamma\Delta \urcorner : (\alpha\Gamma\Delta(\!|\theta|\!)B_c)^{\sigma_c}$ (with the condition on variables: by Lemma 3.16 $fv(X_l) \subseteq fv(X)$; moreover X appears in $\overrightarrow{R_r}$, and thus $fv(X_l) \subseteq fv(\overrightarrow{R_r})$). Write X_c the resulting term $\ulcorner \alpha\Gamma\Delta \urcorner : (\alpha\Gamma\Delta(\!|\theta|\!)B_c)^{\sigma_c}$. Induction hypothesis gave $B_r \overrightarrow{\overrightarrow{R_{Br}^*}}_{\mathcal{N}} B_c$ with $\theta \# \overrightarrow{R_{Br}^*}$, thus by Lemma 3.17 there is a sequence $\overrightarrow{R_{Br}^{**}}$ with $fv(\overrightarrow{R_{Br}^{**}}) = fv(\overrightarrow{R_{Br}^*})$ such that $\alpha\Gamma\Delta(\!|\theta|\!)B_r \overrightarrow{\overrightarrow{R_{Br}^{**}}}_{\mathcal{N}} \alpha\Gamma\Delta(\!|\theta|\!)B_c$. Since $P_r \Rightarrow_{\mathcal{N}} P_c$ and $A_r \Rightarrow_{\mathcal{N}} A_c$, by Lemma 3.14 $\sigma_r \Rightarrow_{\mathcal{N}} \sigma_c$, with $dom(\sigma_r) = dom(\sigma_c) = \theta$. Hence by Lemma 3.18 $X_r = \ulcorner \alpha\Gamma\Delta \urcorner : (\alpha\Gamma\Delta(\!|\theta|\!)B_r)^{\sigma_r} \Rightarrow_{\mathcal{N}} \ulcorner \alpha\Gamma\Delta \urcorner : (\alpha\Gamma\Delta(\!|\theta|\!)B_c)^{\sigma_c}$ and reduction $X_r \Rightarrow_{\mathcal{N}} X_c$ also holds (with asked variables inclusion too).
- If $X = T_1 T_2 \Rightarrow_{\mathcal{N}} T_{l1} T_{l2} = X_l$ with $T_1 \Rightarrow_{\mathcal{N}} T_{l1}$ and $T_2 \Rightarrow_{\mathcal{N}} T_{l2}$, then $X \Rightarrow_{\mathcal{N}} T_{r1} T_{r2} = X_r$ with $T_1 \Rightarrow_{\mathcal{N}} T_{r1}$ and $T_2 \Rightarrow_{\mathcal{N}} T_{r2}$. By induction hypothesis $T_{l1} \Rightarrow_{\mathcal{N}} T_{c1} \; \mathcal{N} \Leftarrow T_{r1}$ and $T_{l2} \Rightarrow_{\mathcal{N}} T_{c2} \; \mathcal{N} \Leftarrow T_{r2}$ with asked conditions on variables. Conclusion is straightforward.
- If $X = [\theta]\, P \to B \Rightarrow_{\mathcal{N}} [\theta]\, P_l \to B_l = X_l$ with $P \Rightarrow_{\mathcal{N}} P_l$ and $B \overrightarrow{\overrightarrow{R_{Bl}}}_{\mathcal{N}} B_l$ and $\theta \# \overrightarrow{R_{Bl}}$, then $X \Rightarrow_{\mathcal{N}} [\theta]\, P_r \to B_r$ with $P \Rightarrow_{\mathcal{N}} P_r$ with $B \overrightarrow{\overrightarrow{R_{Br}}}_{\mathcal{N}} B_r$ and $\theta \# \overrightarrow{R_{Br}}$. By induction hypothesis $P_l \Rightarrow_{\mathcal{N}} P_c \; \mathcal{N} \Leftarrow P_r$, $B_l \overrightarrow{\overrightarrow{R_{Bl}^*}}_{\mathcal{N}} B_c \; \mathcal{N} \overleftarrow{\overrightarrow{R_{Br}^*}} B_r$, $\theta \# \overrightarrow{R_{Bl}^*}$ and $\theta \# \overrightarrow{R_{Bl}^*}$, which allows to derive the conclusion.

- If $X = \alpha : ((\Gamma \cdot [\theta]\, P \to B)A) \Rightarrow X_l$ with `fail_3.15` rule, then there are three cases:
  - If $X = \alpha : Z \Rightarrow_{\mathcal{N}} \alpha : Z_r = X_r$ with $Z \Rightarrow_{\mathcal{N}} Z_r$, see the already solved symmetrical case.
  - If $X \Rightarrow_{\mathcal{N}} X_r$ also with `fail_3.15` rule. Then $X_l = \ulcorner \alpha\Gamma\Delta \urcorner : \bot = X_r$. Then for $X_c = X_l = X_r$, the reductions $X_l \overset{\emptyset}{\Rightarrow}_{\mathcal{N}} X_c \,{}_{\mathcal{N}}\overset{\emptyset}{\Leftarrow} X_r$ complete the diamond.
  - The case $X \Rightarrow_{\mathcal{N}} X_r$ with `substitution_3.15` rule is impossible. It would imply $\{A/[\theta]\, P\} = (\Delta, \mu)$ with μ a substitution. But here $\mu = \bot$.
- If $X = \alpha : ((\Gamma \cdot [\theta]\, P \to B)A) \Rightarrow X_l$ with `substitution_3.15` rule, then there are still three cases:
  - If $X = \alpha : Z \Rightarrow_{\mathcal{N}} \alpha : Z_r = X_r$ with $Z \Rightarrow_{\mathcal{N}} Z_r$, see the already solved symmetrical case.
  - As above, the case $X \Rightarrow_{\mathcal{N}} X_r$ with `fail_3.15` rule is impossible.
  - If $X \Rightarrow_{\mathcal{N}} X_r$ with `substitution_3.15` rule. Summary of the case:
    $\{A/[\theta]\, P\} = (\Delta, \sigma)$, $P_l \,{}_{\mathcal{N}}\Leftarrow P \Rightarrow_{\mathcal{N}} P_r$, $A_l \,{}_{\mathcal{N}}\Leftarrow A \Rightarrow_{\mathcal{N}} A_r$, $B_l \,{}_{\mathcal{N}}\overset{\overrightarrow{R_{Bl}}}{\Longleftarrow} B \overset{\overrightarrow{R_{Br}}}{\Longrightarrow}_{\mathcal{N}} B_r$ with $\theta\#\overrightarrow{R_{Bl}}$ and $\theta\#\overrightarrow{R_{Br}}$, $\{A_l/[\theta]\, P_l\} = (\Delta, \sigma_l)$ and $\{A_r/[\theta]\, A_r\} = (\Delta, \sigma_r)$. And finally $X_l = \ulcorner \alpha\Gamma\Delta \urcorner : (\alpha\Gamma\Delta(\!|\theta|\!)B_l)^{\sigma_l}$ and $X_r = \ulcorner \alpha\Gamma\Delta \urcorner : (\alpha\Gamma\Delta(\!|\theta|\!)B_r)^{\sigma_r}$.
    By induction hypothesis $P_l \Rightarrow_{\mathcal{N}} P_c \,{}_{\mathcal{N}}\Leftarrow P_r$, $A_l \Rightarrow_{\mathcal{N}} A_c \,{}_{\mathcal{N}}\Leftarrow A_r$, and then by Lemma 3.14 $\sigma_l \Rightarrow_{\mathcal{N}} \sigma_c \,{}_{\mathcal{N}}\Leftarrow \sigma_r$. Induction hypothesis also gives and $B_l \overset{\overrightarrow{R^*_{Bl}}}{\Longrightarrow}_{\mathcal{N}} B_c \,{}_{\mathcal{N}}\overset{\overrightarrow{R^*_{Br}}}{\Longleftarrow} B_r$ with $\theta\#\overrightarrow{R^*_{Bl}}$ and $\theta\#\overrightarrow{R^*_{Br}}$. By Lemma 3.17, $\alpha\Gamma\Delta(\!|\theta|\!)B_l \overset{\overrightarrow{R^*_{Bl}}}{\Longrightarrow}_{\mathcal{N}} \alpha\Gamma\Delta(\!|\theta|\!)B_c \,{}_{\mathcal{N}}\overset{\overrightarrow{R^*_{Br}}}{\Longleftarrow} \alpha\Gamma\Delta(\!|\theta|\!)B_r$. Moreover $dom(\sigma_r) = dom(\sigma_l) = dom(\sigma) = \theta$. Then by Lemma 3.18, $(\alpha\Gamma\Delta(\!|\theta|\!)B_l)^{\sigma_l} \overset{\overrightarrow{R^{**}_{Bl}}}{\Longrightarrow}_{\mathcal{N}} (\alpha\Gamma\Delta(\!|\theta|\!)B_c)^{\sigma_c} \,{}_{\mathcal{N}}\overset{\overrightarrow{R^{**}_{Br}}}{\Longleftarrow} (\alpha\Gamma\Delta(\!|\theta|\!)B_r)^{\sigma_r}$, and finally $X_l \overset{\overrightarrow{R^{**}_{Bl}}}{\Longrightarrow}_{\mathcal{N}} \ulcorner \Gamma \urcorner : (\alpha\Gamma\Delta(\!|\theta|\!)B_c)^{\sigma_c} \,{}_{\mathcal{N}}\overset{\overrightarrow{R^{**}_{Br}}}{\Longleftarrow} X_r$. □

Lemma 3.20 (Simulation). $\to_{\mathcal{N}} \subseteq \Rightarrow_{\mathcal{N}} \subseteq \to^*_{\mathcal{N}}$

Proof.

- $\overset{R}{\to}_{\mathcal{N}} \subseteq \overset{R}{\Rightarrow}_{\mathcal{N}}$, by induction on $\overset{R}{\to}_{\mathcal{N}}$. For instance: if $[\theta]\, P \to B \overset{R}{\to}_{\mathcal{N}} [\theta]\, P \to B'$ with $B \overset{R}{\to}_{\mathcal{N}} B'$ and $\theta\#R$. By induction hypothesis $B \overset{R}{\Rightarrow}_{\mathcal{N}} B'$. Moreover $P \overset{\emptyset}{\Rightarrow}_{\mathcal{N}} P'$. Then $[\theta]\, P \to B \overset{\emptyset;R}{\Longrightarrow}_{\mathcal{N}} [\theta]\, P \to B'$, which is equal to $[\theta]\, P \to B \overset{R}{\Rightarrow}_{\mathcal{N}} [\theta]\, P \to B$. Other cases are similar.
- For any reduction $T \overset{R_1 \ldots R_n}{\Longrightarrow}_{\mathcal{N}} T'$, there exists an integer $m \geq n$, a sequence $R^*_1 \ldots R^*_m$, a function $\varphi : \{1 \ldots m\} \to \{1 \ldots n\}$ such that for all i, $fv(R^*_{\varphi(i)}) \subseteq fv(R_i)$, and $T \overset{R^*_1}{\longrightarrow}_{\mathcal{N}} \ldots \overset{R^*_n}{\longrightarrow}_{\mathcal{N}} T'$, by induction on $\overset{\overrightarrow{R}}{\Longrightarrow}_{\mathcal{N}}$.
  For instance, suppose $T = \alpha : ((\Gamma \cdot [\theta]\, P \to B)A) \overset{\overrightarrow{T R_P R_B R_A}}{\Longrightarrow}_{\mathcal{N}} \ulcorner \alpha\Gamma\Delta \urcorner : (\alpha\Gamma\Delta(\!|\theta|\!)B')^{\sigma'} = T'$ by `substitution_3.15` rule, with $B \overset{\overrightarrow{R_B}}{\Longrightarrow}_{\mathcal{N}} B'$ (and $\theta\#\overrightarrow{R_B}$), $\{A/[\theta]\, P\} = (\Delta, \sigma)$, $P \overset{\overrightarrow{R_P}}{\Longrightarrow}_{\mathcal{N}} P'$, $A \overset{\overrightarrow{R_A}}{\Longrightarrow}_{\mathcal{N}} A'$, $\{A'/[\theta]\, P'\} = (\Delta, \sigma')$ and $\alpha\Gamma\Delta \in \mathcal{N}$. By induction hypothesis, there are $\overrightarrow{R^*_P}, \overrightarrow{R^*_B}, \overrightarrow{R^*_A}$ such that $P \overset{\overrightarrow{R^*_P}}{\longrightarrow}_{\mathcal{N}} P'$, $B \overset{\overrightarrow{R^*_B}}{\longrightarrow}_{\mathcal{N}} B'$ and $A \overset{\overrightarrow{R^*_A}}{\longrightarrow}_{\mathcal{N}} A'$ (with functions

mapping elements of $\overrightarrow{R^*_Z}$ on elements of $\overrightarrow{R_Z}$ for any Z in $\{A, B, P\}$). Since $\theta\#\overrightarrow{R_B}$ and $fv(\overrightarrow{R^*_B}) \subseteq fv(\overrightarrow{R_B})$, $\theta\#\overrightarrow{R^*_B}$. Then $[\theta]\, P \to B \xrightarrow{\overrightarrow{R^*_P}}_{\mathcal{N}} [\theta]\, P' \to B \xrightarrow{\overrightarrow{R^*_B}}_{\mathcal{N}} [\theta]\, P' \to B'$ and

$$\begin{array}{rll} \alpha : ((\Gamma \cdot [\theta]\, P \to B)A) & \xrightarrow{\overrightarrow{R^*_P}\overrightarrow{R^*_B}}_{\mathcal{N}} & \alpha : ((\Gamma \cdot [\theta]\, P' \to B')A) \\ & \xrightarrow{\overrightarrow{R^*_A}}_{\mathcal{N}} & \alpha : ((\Gamma \cdot [\theta]\, P' \to B')A') = T'' \end{array}$$

Moreover, the name of the redex T'' is $\alpha\Gamma\Delta \in \mathcal{N}$, and thus

$$T'' \xrightarrow{T''}_{\mathcal{N}} \ulcorner \alpha\Gamma\Delta' \urcorner : (\alpha\Gamma\Delta' (\!|\theta|\!) B')^{\sigma'} = T'$$

Remark also that $T \Rightarrow_{\mathcal{N}} T''$, and hence by Lemma 3.16 $fv(T'') \subseteq fv(T)$.
Other cases are similar. □

Theorem 3.21 (**Parametric Confluence**). *For any set of names $\mathcal{N}$, the reduction relation $\to_{\mathcal{N}}$ is confluent.*

Proof. Reduction relation $\Rightarrow_{\mathcal{N}}$ has the diamond property (Diamond Lemma 3.19), thus $\Rightarrow^*_{\mathcal{N}}$ has the diamond property. Moreover, Simulation Lemma 3.20 gives $\to_{\mathcal{N}} \subseteq \Rightarrow_{\mathcal{N}} \subseteq \to^*_{\mathcal{N}}$. Hence $\to^*_{\mathcal{N}} \subseteq \Rightarrow^*_{\mathcal{N}} \subseteq (\to^*_{\mathcal{N}})^*$. But $(\to^*_{\mathcal{N}})^* = \to^*_{\mathcal{N}}$, and thus $\to^*_{\mathcal{N}} \subseteq \Rightarrow^*_{\mathcal{N}} \subseteq \to^*_{\mathcal{N}}$. Finally $\Rightarrow^*_{\mathcal{N}} = \to^*_{\mathcal{N}}$. Hence $\to^*_{\mathcal{N}}$, as $\Rightarrow^*_{\mathcal{N}}$, has the diamond property, and $\to_{\mathcal{N}}$ is confluent. □

3.3.4 Termination Theorem

Lemma 3.22. *Let $\mathcal{N}$ be a set of labels and consider restricted reduction $\to_{\mathcal{N}}$. For any labelled term T and any label α, termination of T implies termination of $\alpha : T$.*

Proof. By Lemma 3.5, residuals of T are labelled terms. Thus residuals of $\alpha : T$ have the form $\alpha : \beta_r : X_r$ and there is no reduction at the root: T and $\alpha : T$ have the same reductions. □

Lemma 3.23. *Let $\mathcal{N}$ be a finite set of labels, and consider restricted reduction $\to_{\mathcal{N}}$. Let T be a terminating labelled term, σ a terminating substitution of domain θ, and Ω a label.*
Then $(\Omega(\!|\theta|\!)T)^\sigma$ is terminating.

Proof. Denote by $\mathbb{R}$ the lexicographic product of $\succ$ (reversed direct contribution) and subterm order. Choose a pair (Ω, T) with $\Omega \in \mathcal{N}$ and T terminating, minimal for $\mathbb{R}$ such that there exists a terminating subsitution σ making $(\Omega(\!|\theta|\!)T)^\sigma$ non-terminating (θ denotes the domain of σ). Such a pair exists by finiteness of $\mathcal{N}$. Note $T = \alpha : X$. Contradiction is shown by cases on X. First remark that if $\theta\#X$ then $(\Omega(\!|\theta|\!)X)^\sigma = X$, which is terminating by hypothesis. Then suppose $\theta \cap fv(X) \neq \emptyset$. Hence $(\Omega(\!|\theta|\!)T)^\sigma = [\Omega, \alpha] : (\Omega(\!|\theta|\!)X)^\sigma$.

- If X is a labelled term, then by Lemma 3.22 $(\Gamma(\!|\theta|\!)X)^\sigma$ alone is non-terminating, which contradicts minimality.
- If $X = x$ with $x \notin \theta$ or $X = \hat{x}$, then $\theta\#X$, which is not the case.
- If $X = x$ with $x \in \theta$, then $(\Omega(\!|\theta|\!)X)^\sigma = \sigma_x$. Conclusion is by termination of σ (in particular of σ_x) and Lemma 3.22 (since σ_x is a labelled term).
- If $X = [\tau]\, P \to B$ with $\theta\#\tau$, then $(\Omega(\!|\theta|\!)X)^\sigma = [\tau]\, (\Omega(\!|\theta|\!)P)^\sigma \to (\Omega(\!|\theta|\!)B)^\sigma$. Since P and B are labelled subterms of T, by minimality $(\Omega(\!|\theta|\!)P)^\sigma$ and $(\Omega(\!|\theta|\!)B)^\sigma$ are terminating, and so does $(\Omega(\!|\theta|\!)T)^\sigma$.

- If $X = T_1T_2$, then $(\Omega(\!|\theta|\!)X)^\sigma = (\Omega(\!|\theta|\!)T_1)^\sigma(\Omega(\!|\theta|\!)T_2)^\sigma$, as in the precedent case, $(\Omega(\!|\theta|\!)T_1)^\sigma$ and $(\Omega(\!|\theta|\!)T_2)^\sigma$ are terminating. Hence any infinite reduction will eventually reduce a redex at the root :

$$\begin{array}{rl} & [\Omega, \alpha] : ((\Omega(\!|\theta|\!)T_1)^\sigma(\Omega(\!|\theta|\!)T_2)^\sigma) \\ \xrightarrow{>\epsilon}{}^* & [\Omega, \alpha] : ((\Gamma_1 : [\theta_1]\, P_1 \to B_1)A_2) \\ \xrightarrow{\epsilon} & \ulcorner\Omega^*\urcorner : (\Omega^*(\!|\theta_1|\!)B_1)^{\mu_2} \end{array}$$

  where $\xrightarrow{\epsilon}$ is a reduction at the root, and $\xrightarrow{>\epsilon}$ a reduction anywhere else. If $\mu_2 = \bot$, then the result is a terminating term. Suppose μ_2 is a substitution σ_2. Ω^* has the form $[\Omega, \alpha]\,\Gamma_1\Delta_0$, hence $\Omega^* \succ \Omega$. Moreover B_1 and A_2 are terminating (they are reduced subterms of T_1 and T_2). Then σ_2 is also terminating since all substitutes are subterms of A_2. Hence by minimality $(\Omega^*(\!|\theta_1|\!)B_1)^{\sigma_2}$ is terminating, and by Lemma 3.22 $\ulcorner\Omega^*\urcorner : (\Omega^*(\!|\theta_1|\!)B_1)^{\sigma_2}$ is also terminating. □

Theorem 3.24 (Termination Theorem)**.** *For any finite set of labels $\mathcal{N}$, $\to_{\mathcal{N}}$ is strongly normalizing.*

Proof. Let X be a term. By induction on X, there is no infinite reduction starting from X.

- Case $X = x$ or $X = \hat{x}$: X is a normal form.
- Case $X = [\theta]\, P \to B$. Any reduction of X is a reduction of P or B. But by induction hypothesis any reduction from P or B is finite.
- Case $X = T_1T_2$. Any reduction of X is a reduction of T_1 or T_2. But by induction hypothesis any reduction from T_1 or T_2 is finite.
- Case $X = \alpha : Z$. By induction hypothesis any reduction from Z is terminating. Suppose there is an infinite reduction from X, it eventually reduces X at the root:

$$\begin{array}{rl} & \alpha : Z \\ \xrightarrow{>\epsilon}{}^* & \alpha : ((\Gamma : [\theta]\, P \to B)A) \\ \xrightarrow{\epsilon} & \ulcorner\Omega\urcorner : (\Omega(\!|\theta|\!)B)^{\mu} \end{array}$$

  If $\mu = \bot$, then the result is $\ulcorner\Omega\urcorner : \bot$, which is a normal form. Suppose μ is a substitution σ. A and B are subterms of a residual of Z: they are terminating. Since for any $x \in dom(\sigma)$, σ_x is a subterm of B, σ is also terminating. Hence by Lemma 3.23, $(\Omega(\!|\theta|\!)B)^\sigma$ is terminating. □

3.3.5 Finite Developments Theorem

In this section, when $\mathcal{R}$ denotes a set of redexes, the symbol is also used to denote the set of positions of these redexes. The same shortcut is used with $\mathcal{S}$ for subterms.

Theorem 3.25. *For any term t of* WPPC*, any set of redexes $\mathcal{R}$, and any set of subterms $\mathcal{S}$:*

- *Any development of $\mathcal{R}$ is finite.*
- *All complete developments of $\mathcal{R}$ yield the same result.*
- *The set of residuals of $\mathcal{S}$ after any complete development of $\mathcal{R}$ is the same.*

Proof. Particular case of Finite Developments Theorem 3.32 for *LWPPC*. □

The proof of Finite Development Theorem 3.32 for *LWPPC* is deduced from Termination Theorem 3.24 and Confluence Theorem 3.12 through the following trick: to mark redexes to be developped and subterms whose residuals are interesting, the internal labelling system of *LWPPC* is used by introducing some fresh initial labels (notions of marking and marking functions below). Intermediate steps are a characterization of residuals in the marked terms (Lemma 3.29) and equivalences between developments in the original term and already known reductions such as $\rightarrow_{\mathcal{N}}$ (Lemma 3.31). A **morphism** φ is a function from initial labels to labels. By extension, it applies to terms and labels in the following way:

$$\begin{array}{rcl@{\qquad}rcl}
\varphi(x) & := & x & \varphi(\mathbb{o}) & := & \varphi_{\mathbb{o}} \\
\varphi(\hat{x}) & := & \hat{x} & \varphi(\ulcorner\Omega\urcorner) & := & \ulcorner\varphi(\Omega)\urcorner \\
\varphi(T_1T_2) & := & \varphi(T_1)\varphi(T_2) & \varphi([\Omega,\alpha]) & := & [\varphi(\Omega),\varphi(\alpha)] \\
\varphi([\theta]\,P \rightarrow B) & := & [\theta]\,\varphi(P) \rightarrow \varphi(B) & \varphi(\alpha_1...\alpha_n) & := & \varphi(\alpha_1)...\varphi(\alpha_n) \\
\varphi(\alpha : Z) & := & \varphi(\alpha) : \varphi(Z) & & &
\end{array}$$

Lemma 3.26. *For any morphism φ and reduction $X \xrightarrow{R}_{\mathcal{N}} X'$, $\varphi(X) \xrightarrow{\varphi(R)}_{\varphi(\mathcal{N})} \varphi(X')$.*

Proof. By induction on $X \xrightarrow{R}_{\mathcal{N}} X'$. □As a consequence, morphisms can also be applied to reductions: for any reduction $\overrightarrow{\rho} = X_0 \xrightarrow{R_1}_{\mathcal{N}_1} X_1 \xrightarrow{R_2}_{\mathcal{N}_2} ... \xrightarrow{R_n}_{\mathcal{N}_n} X_n$, $\varphi(\overrightarrow{\rho})$ is the reduction $\varphi(X_0) \xrightarrow{\varphi(R_1)}_{\varphi(\mathcal{N}_1)} \varphi(X_1) \xrightarrow{\varphi(R_2)}_{\varphi(\mathcal{N}_2)} ... \xrightarrow{\varphi(R_n)}_{\varphi(\mathcal{N}_n)} \varphi(X_n)$, whose existence is given by Lemma 3.26.

Lemma 3.27. *Let X be a term, and φ a morphism. Let $\mathfrak{p}, \mathfrak{q}$ be positions of X. Then $X|_{\mathfrak{p}}$ is a redex R if and only if $\varphi(X)|_{\mathfrak{p}}$ is a redex. In this case, note ρ the reduction of $X|_{\mathfrak{p}}$. Then $\mathfrak{q}/\rho = \mathfrak{q}/\varphi(\rho)$ ($\mathfrak{q}/\rho$ denotes the set of positions of residuals of the subterm at position $\mathfrak{q}$).*

Proof. First note that forall Y, Y is matchable if and only if $\varphi(Y)$ is matchable, and in this case $|\varphi(Y)| = \varphi(|Y|)$. Deduce that for any Y, Z, $\{\!\{Y/[\theta]\,Z\}\!\}_p$ is defined if and only if $\{\!\{\varphi(Y)/[\theta]\,\varphi(Z)\}\!\}_p$ is defined, and in this case $\{\!\{\varphi(Y)/[\theta]\,\varphi(Z)\}\!\}_p = \varphi(\{\!\{Y/[\theta]\,Z\}\!\}_p)$. Moreover for any Y the set of positions of Y and the set of positions of $\varphi(Y)$ are equal. □

Lemma 3.28. *For any morphism and any reduction $\overrightarrow{\rho} : \varphi(X) \rightarrow X'$, there is a reduction $\overrightarrow{\rho^*}$ such that $\varphi(\overrightarrow{\rho^*}) = \overrightarrow{\rho}$.*

Proof. By induction on the length of $\overrightarrow{\rho}$.

- If $\overrightarrow{\rho}$, then take $\overrightarrow{\rho^*}$ the empty reduction on X.
- If $\overrightarrow{\rho} = \overrightarrow{\rho_0}\rho_n$, with $\overrightarrow{\rho_0} : \varphi(X) \rightarrow X_1$ and $\rho_n : X_1 \xrightarrow{R} X'$. Let $\mathfrak{p}$ be the position of R in X_1. By induction hypothesis there is $\overrightarrow{\rho_0^*} : X \xrightarrow{R} X_1^*$ with $\varphi(\overrightarrow{\rho_0^*}) = \overrightarrow{\rho_0}$. Then $\varphi(X_1^*) = X_1$, and by Lemma 3.27 $X_1^*|_{\mathfrak{p}}$ is a redex R^* such that $\rho_n^* : X_1^* \xrightarrow{R^*} (X')^*$ satisfies $\varphi(\rho_n^*) = \rho_n$. □

Let X be a term, $\mathcal{P}$ a set of positions such that $X|_{\mathfrak{p}}$ is a labelled term for any $\mathfrak{p} \in \mathcal{P}$, and f a function from $\mathcal{P}$ to the set of initial labels. The **marking** of X via f is the pair $(X^\bullet, \varphi)$ such that:

- $X^\bullet$ is the term X where for all $\mathfrak{p} \in \mathcal{P}$, label at position $\mathfrak{p}$ is replaced by $f(\mathfrak{p})$.
- φ is the morphism defined by mapping of $f(\mathfrak{p})$ to $\alpha_{\mathfrak{p}}$ for all $\mathfrak{p} \in \mathcal{P}$.

Remark that in this case, $\varphi(X^\bullet) = X$. Let X be a term, and $\mathcal{P}$ a set of positions of X. If X is a clipped term, suppose $\epsilon \notin \mathcal{P}$. Decompose $\mathcal{P}$ as the disjoint union of $\mathcal{P}_l$ and $\mathcal{P}_c$ such that $X|_\mathfrak{p}$ is a labelled subterm for all $\mathfrak{p} \in \mathcal{P}_l$, and $X|_\mathfrak{p}$ is a clipped strict subterm of X for all $\mathfrak{p} \in \mathcal{P}_c$. For all $\mathfrak{p} \in \mathcal{P}_c$, $\mathfrak{p} = \mathfrak{q}\mathbb{z}$ (since $\epsilon \notin \mathcal{P}_c$, and by inspection of the grammar of terms), which means $X|_\mathfrak{q}$ is a labelled term. Note $\mathcal{P}_c^- = \{\mathfrak{q} | \mathfrak{q}\mathbb{z} \in \mathcal{P}_c\}$. A **marking function** of $\mathcal{P}$ is an injective function f from $\mathcal{P}_l \cup \mathcal{P}_c^-$ to the set of initial labels, such that for any $\mathfrak{p} \in \mathcal{P}_l \cup \mathcal{P}_c^-$, $f(\mathfrak{p})$ is a fresh initial label.

For any atomic label α, denote by $\iota[\alpha]$ the set of labels generated by the following conditions (*i.e.* the smallest set satisfying the conditions):

- $\alpha \in \iota[\alpha]$
- For any $\beta \in \iota[\alpha]$ and any label Ω, $[\Omega, \beta] \in \iota[\alpha]$.

Lemma 3.29. *Let X be a term, $\mathcal{P}$ a set of positions of X (with $\epsilon \notin \mathcal{P}$ if X is a clipped term), and f a marking function of $\mathcal{P}$. Write $(X^\bullet, \varphi)$ the marking of X via f, and suppose there is a reduction $\rho : X^\bullet \xrightarrow{R_1}_\mathcal{N} ... \xrightarrow{R_n}_\mathcal{N} X'$. Then:*

- *For any $\mathfrak{p} \in \mathcal{P}_l$, residuals of $X^\bullet|_\mathfrak{p}$ are exactly subterms $\alpha' : Z'$ of X' such that $\alpha' \in \iota[f(\mathfrak{p})]$.*
- *For any $\mathfrak{p} \in \mathcal{P}_c$, residuals of $X^\bullet|_\mathfrak{p}$ are exactly the clipped subterms N' of X' occurring as $\alpha' : N'$ with $\alpha' \in \iota[f(\mathfrak{q})]$ (for $\mathfrak{p} = \mathfrak{q}\mathbb{z}$).*

Proof.

- Let $\mathfrak{p} \in \mathcal{P}_l$. By Lemma 3.5, residuals of $X^\bullet|_\mathfrak{p}$ have the asked form.
  Conversely, let $\alpha' : Z'$ be a subterm of X' with $\alpha' \in \iota[f(\mathfrak{p})]$. Since $f(\mathfrak{p})$ is an initial label, by Lemma 3.6 $\alpha' : Z'$ has an ancestor $\alpha^\bullet : Z^\bullet$ in $X^\bullet$, and this ancestor satisfies $\alpha^\bullet \in \iota[f(\mathfrak{p})]$. But label $f(\mathfrak{p})$ had been taken fresh in X, with f injective. Thus label $f(\mathfrak{p})$ has a unique occurrence in $X^\bullet$, which is $X^\bullet|_\mathfrak{p}$.
- Proof of the second point is by combination of the first point and Lemma 3.7. □

Corollary 3.30. *With the hypothesis of Lemma 3.29, if $X|_\mathfrak{p}$ is a redex (for $\mathfrak{p} \in \mathcal{P}$), then its residuals are exactly the subterms of X' with label $f(\mathfrak{p})$.*

Proof. Apply Lemma 3.10 which ensures that the residual of a redex is a redex with same name, and thus with same label at the root. □

Lemma 3.31. *Let X be a term, $\mathcal{R}$ a set of redexes of X, $\mathcal{S}$ a set of subterms of X (strict subterms if X is a clipped terms), f a marking function of $\mathcal{R} \cup \mathcal{S}$. Write $(X^\bullet, \varphi)$ the marking of X via f, and $\mathcal{N}^\bullet$ the set of names of redexes of $\mathcal{R}$ in $X^\bullet$. Then there is a development $\overrightarrow{\rho} : X \xrightarrow{R_1} X_1 \xrightarrow{R_2} ... \xrightarrow{R_n} X_n$ of $\mathcal{R}$ if and only if there is a reduction $\overrightarrow{\rho^\bullet} : X^\bullet \xrightarrow{R_1^\bullet}_{\mathcal{N}^\bullet} X_1^\bullet \xrightarrow{R_2^\bullet}_{\mathcal{N}^\bullet} ... \xrightarrow{R_n^\bullet}_{\mathcal{N}^\bullet} X_n^\bullet$ with $\varphi(X_i^\bullet) = X_i$ and $\varphi(R_i^\bullet) = R_i$ for any $i \in \{1...n\}$.*

Proof. Proof is in two steps:

- There is a development $\overrightarrow{\rho}$ of $\mathcal{R}$ in X if and only if there is a development $\overrightarrow{\rho^\bullet}$ of $\mathcal{R}$ in $X^\bullet$ such that $\varphi(\overrightarrow{\rho^\bullet}) = \overrightarrow{\rho}$.
  Direct implication, by induction on the length of $\overrightarrow{\rho}$:
  - If $\overrightarrow{\rho}$ is the empty reduction, take $\overrightarrow{\rho^\bullet}$ as the empty reduction from $X^\bullet$.

– If $\overrightarrow{\rho} = \overrightarrow{\rho_0}\rho_n$, by induction there is a development $\overrightarrow{\rho_0^\bullet} : X^\bullet \to X^\bullet_{n-1}$ of $\mathcal{R}$ in $X^\bullet$ with $\varphi(\overrightarrow{\rho_0^\bullet}) = \overrightarrow{\rho_0}$. In particular $\varphi(X^\bullet_{n-1}) = X_{n-1}$. Thus by Lemma 3.28 there is a reduction $\rho_n^\bullet$ such that $\varphi(\rho_n^\bullet) = \rho_n$.

Reverse implication: for any development $\overrightarrow{\rho^\bullet}$ of $\mathcal{R}$ in $X^\bullet$, $\varphi(\overrightarrow{\rho^\bullet})$ is a development of $\mathcal{R}$ in X, by induction on the length of $\overrightarrow{\rho^\bullet}$.

– If $\overrightarrow{\rho^\bullet}$ is the empty reduction, conclusion is immediate.
– If $\overrightarrow{\rho^\bullet} = \overrightarrow{\rho_0^\bullet}\rho_n^\bullet$, by induction $\varphi(\overrightarrow{\rho_0^\bullet})$ is a development of $\mathcal{R}$ in X. Since $\overrightarrow{\rho^\bullet}$ is a development of $\mathcal{R}$, the redex $R_n^\bullet$ reduced by $\rho_n^\bullet$ is a residual of some redex $R^\bullet$ of $\mathcal{R}$ (in $X^\bullet$). Then by Lemma 3.27, the redex $\varphi(R_n^\bullet)$ reduced by $\varphi(\rho_n^\bullet)$ is a residual of some redex R of $\mathcal{R}$ (in X). Hence $\varphi(\overrightarrow{\rho^\bullet})$ is a development of $\mathcal{R}$ in X.

- Developments of $\mathcal{R}$ in $X^\bullet$ are exactly the reductions from $X^\bullet$ restrained by $\mathcal{N}^\bullet$.
Let $\overrightarrow{\rho^\bullet} = \rho_1^\bullet ... \rho_n^\bullet$ be a development of $\mathcal{R}$ in $X^\bullet$. By definition of a development, for all $i \in \{1...n\}$ $\rho_i^\bullet \in \mathcal{R}/(\rho_1^\bullet ... \rho_{i-1}^\bullet)$. By Lemma 3.10, the name $\Omega_i^\bullet$ of the redex $R_i^\bullet$ reduced by $\rho_i^\bullet$ is the name of its ancestor in $X^\bullet$. Thus $\Omega_i^\bullet \in \mathcal{N}^\bullet$.
Conversely, let $\overrightarrow{\rho_\bullet} = \rho_\bullet^1 ... \rho_\bullet^m$ be a sequence of reduction from $X^\bullet$ for $\to_{\mathcal{N}^\bullet}$. Let $R_\bullet^i$ be the redex reduced by $\rho_\bullet^i$, and $\Omega_\bullet^i$ its name. For any i, $\Omega_\bullet^i \in \mathcal{N}^\bullet$. In particular, there is a position $\mathfrak{p}_i$ of a redex of $X^\bullet$ such that $\mathfrak{p}_i \in \mathcal{R}$ and $\Omega_\bullet^i = f(\mathfrak{p}_i)\Omega_0^i$, and $R_\bullet^i$ has the form $f(\mathfrak{p}_i) : Z_\bullet^i$. By Corollary 3.30, $R_\bullet^i$ is a residual of $X^\bullet|_{\mathfrak{p}_i}$ after $\rho_\bullet^1 ... \rho_\bullet^{i-1}$. Hence $\overrightarrow{\rho_\bullet}$ is a development of $\mathcal{R}$ in $X^\bullet$. □

Theorem 3.32 (Finite Developments Theorem)**.** *For any term X, any set $\mathcal{R}$ of redexes of X, and any set $\mathcal{S}$ of subterms of X:*

- *Any development of $\mathcal{R}$ is finite.*
- *All complete developments of $\mathcal{R}$ yield the same result.*
- *The set of residuals of $\mathcal{S}$ after any complete development of $\mathcal{R}$ is the same.*

Proof. Suppose X is a labelled term or $\epsilon \in \mathcal{S}$. Let f be a marking function of $\mathcal{R} \cup \mathcal{S}$. Write $(X^\bullet, \varphi)$ the marking of X via f and $\mathcal{N}^\bullet$ the set of names of redexes of $\mathcal{R}$ in $X^\bullet$.

- Remark that $\mathcal{N}^\bullet$ is finite since X is finite. Thus by Termination Theorem 3.24 $\to_{\mathcal{N}^\bullet}$ is strongly normalizing. Let $\overrightarrow{\rho}$ be a development of $\mathcal{R}$ in X. By Lemma 3.31, there is a reduction $\overrightarrow{\rho^\bullet}$ of $\to_{\mathcal{N}^\bullet}$ in $X^\bullet$ (necessarily finite) with $\varphi(\overrightarrow{\rho^\bullet}) = \overrightarrow{\rho}$. Then $\overrightarrow{\rho}$ is finite.

- By Parametric Confluence Theorem 3.21 $\to_{\mathcal{N}^\bullet}$ is confluent. Let $\overrightarrow{\rho_1}$ and $\overrightarrow{\rho_2}$ be two complete developments of $\mathcal{R}$ in X. By Lemma 3.31 there are reductions $\overrightarrow{\rho_1^\bullet}$ and $\overrightarrow{\rho_2^\bullet}$ of $\to_{\mathcal{N}^\bullet}$ in $X^\bullet$ with $\varphi(\overrightarrow{\rho_1^\bullet}) = \overrightarrow{\rho_1}$ and $\varphi(\overrightarrow{\rho_2^\bullet}) = \overrightarrow{\rho_2}$. By confluence of $\to_{\mathcal{N}^\bullet}$ there are reductions $\overrightarrow{\rho_{1+}^\bullet}$ and $\overrightarrow{\rho_{2+}^\bullet}$ of $\to_{\mathcal{N}^\bullet}$ such that $\overrightarrow{\rho_1^\bullet}\overrightarrow{\rho_{1+}^\bullet}$ and $\overrightarrow{\rho_2^\bullet}\overrightarrow{\rho_{2+}^\bullet}$ end at the same term. But by Lemma 3.31, $\varphi(\overrightarrow{\rho_1^\bullet}\overrightarrow{\rho_{1+}^\bullet})$ and $\varphi(\overrightarrow{\rho_2^\bullet}\overrightarrow{\rho_{2+}^\bullet})$ are developments of $\mathcal{R}$ in X. Since $\overrightarrow{\rho_1}$ and $\overrightarrow{\rho_2}$ are complete developments, $\overrightarrow{\rho_{1+}^\bullet}$ and $\overrightarrow{\rho_{2+}^\bullet}$ are empty reductions. Hence $\overrightarrow{\rho_1^\bullet}$ and $\overrightarrow{\rho_2^\bullet}$ end at the same term, and so do $\overrightarrow{\rho_1}$ and $\overrightarrow{\rho_2}$.

- Let $\overrightarrow{\rho_1}$ and $\overrightarrow{\rho_2}$ be two complete developments of $\mathcal{R}$ in X. As in previous point, with Parametric Confluence Theorem 3.21 of confluence and Lemma 3.31 of simulation, there are two developments $\overrightarrow{\rho_1^\bullet}$ and $\overrightarrow{\rho_2^\bullet}$ of $\mathcal{R}$ in $X^\bullet$ such that $\varphi(\overrightarrow{\rho_1^\bullet}) = \overrightarrow{\rho_1}$ and $\varphi(\overrightarrow{\rho_2^\bullet}) = \overrightarrow{\rho_2}$. Moreover $\overrightarrow{\rho_1^\bullet}$ and $\overrightarrow{\rho_2^\bullet}$ end at the same term X'. By Lemma 3.29, residuals of $\mathcal{S}$ are characterized

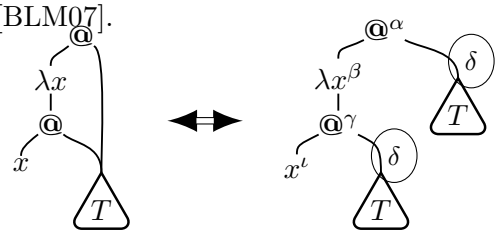
by their labels, which means characterized by X'. Thus $\mathcal{S}/\overrightarrow{\rho_1^\bullet} = \mathcal{S}/\overrightarrow{\rho_2^\bullet}$. By Lemma 3.27, $\mathcal{S}/\varphi(\overrightarrow{\rho_1^\bullet}) = \mathcal{S}/\overrightarrow{\rho_1^\bullet}$ and $\mathcal{S}/\varphi(\overrightarrow{\rho_2^\bullet}) = \mathcal{S}/\overrightarrow{\rho_2^\bullet}$. Hence $\mathcal{S}/\overrightarrow{\rho_1} = \mathcal{S}/\varphi(\overrightarrow{\rho_1^\bullet}) = \mathcal{S}/\overrightarrow{\rho_1^\bullet} = \mathcal{S}/\overrightarrow{\rho_2^\bullet} = \mathcal{S}/\varphi(\overrightarrow{\rho_2^\bullet}) = \mathcal{S}/\overrightarrow{\rho_2}$.

For the last case, if X is a clipped term and $\epsilon \in \mathcal{S}$, apply the same reasoning with $\mathcal{S} \setminus \{\epsilon\}$; remark that X is not a redex and thus that the subterm at position ϵ is always the unique residual of itself. □

LWPPC gives instructions for optimal sharing in *WPPC* by a description of what should be reduced in one single step. But as stated for now these instructions are not constructive. Next section shows how labels can be used to derive an effective graph implementation of sharing.

4 The Sharing Property

Previous section introduces *LWPPC*, a variant of *WPPC* where each term and subterm bears a label (an atomic label or a sequence). The labelling describes a graph implementation with the following idea: each atomic label represents a *memory location*. Subterms with same label are thus meant to be *physically equal*. This idea is specified for first order terms in [Mar92, DLLL05], and for weak λ-calculus in [BLM07].



Of course not all labelled terms can be translated to graphs in a consistent way. The **sharing property** $\mathbb{S}$ is a formal condition allowing it: a term T is said to enjoy the sharing property (noted $\mathbb{S}(T)$) if for any of its subterms $\alpha_1 : Z_1$ and $\alpha_2 : Z_2$,

$$\alpha_1 = \alpha_2 \quad \text{implies} \quad Z_1 = Z_2$$

Main issue is now to check that correspondence with graphs is not only static but also dynamic, by defining a notion of reduction over labelled terms which preserves the sharing property and corresponds to the natural graph reduction.

The **development of a name** $\Rrightarrow$ is defined as follows: given terms T, T', write $T \Rrightarrow T'$ if there exists a label l such that T' is the result of the complete development of the redexes of T with name l (all developments are equivalent by Finite Developments Theorem 3.32).

4.0.1 Direct Contribution Lemma

For labels $\Gamma = \gamma_1...\gamma_n$ and $\Delta = \delta_1...\delta_m$, note $\Gamma \subset \Delta$ when the set inclusion $\{\gamma_1, ..., \gamma_n\} \subseteq \{\delta_1, ..., \delta_m\}$ holds.

This section proves that a redex creating a second redex contributes directly to its name (Direct Contribution Lemma 4.1). First step is to check that compound matching returns the matchability witnesses of the pattern and the argument if needed (Lemmas 4.2 and 4.3). The other intermediate step is to check that a redex creating a matchable form directly contributes to its name, as well as a redex turning an undefined compound matching into a defined compound matching (Lemma 4.4 and 4.6).

Lemma 4.1 (Direct Contribution Lemma)**.** *For any reduction $T \xrightarrow{R} T'$, with redex R of name Ω, if R_c is a redex of T' with name Ω_c created by the reduction, then $\Omega \prec \Omega_c$.*

Proof. Case on redex creation, with Labelled Creation Theorem 3.3:

T1. $R' = \ulcorner\Omega\urcorner : Z'$. Thus $\Gamma_c^2 = \ulcorner\Omega\urcorner\Gamma_c^3$ and $\Gamma_c = \Gamma_c^1\ulcorner\Omega\urcorner\Gamma_c^3$. Hence $\Omega \prec \Omega_c = \alpha_c\Gamma_c\Delta_c$.

T2. ${}'R_c \cap \theta \neq \emptyset$. Thus $R_c = (\Omega(\!|\theta|\!)'R_c)^\sigma = \left[\Omega, \alpha_c^0\right] : Z_c$. Name of R_c is $\Omega_c = \left[\Omega, \alpha_c^0\right]\Gamma_c\Delta_c$ and $\Omega \prec \Omega_c$.

T3. By Lemma 4.6, $\Omega \prec \Delta_c$. Thus $\Omega \prec \Omega_c = \alpha_c\Gamma_c\Delta_c$.

T4. Same basic use of Lemma 4.6. □

Lemma 4.2. *If* $\{\!|Y/[\theta]\,X|\!\}_w = (\Delta, \mu)$ *then* $|Y| \subset \Delta$.

Proof. By induction on $\{\!|Y/[\theta]\,X|\!\}_w$:

- Case $\{\!|Y/[\theta]\,\alpha : Z|\!\}_w = \alpha : \{\!|Y/[\theta]\,Z|\!\}_w$. $\{\!|Y/[\theta]\,Z|\!\}_w = (\Delta_0, \mu)$ with $\Delta = \alpha\Delta_0$. By induction $|Y| \subset \Delta_0$. Then $|Y| \subset \alpha\Delta_0 = \Delta$.
- Cases $X = \hat{x}$ with $x \in \theta$ and $X = [\tau]\,P \to B$: result is (Δ, μ) with $\Delta = |Y|$. Hence $|Y| \subset \Delta$.
- Case $\{\!|\alpha : Z/[\theta]\,X|\!\}_w = \alpha : \{\!|Z/[\theta]\,X|\!\}_w$. $\{\!|Z/[\theta]\,X|\!\}_w = (\Delta_0, \mu)$ with $\Delta = \alpha\Delta_0$. By induction $|Z| \subset \Delta_0$. Then $|Y| = \alpha|Z| \subset \alpha\Delta_0 = \Delta$.
- Case $X = Y = \hat{x}$ with $x \notin \theta$: $|Y| = \varepsilon$ and thus $|Y| \subset \Delta$.
- Case $\{\!|A_1A_2/[\theta]\,P_1P_2|\!\}_w = \{\!|A_1/[\theta]\,P_1|\!\}_w \Subset \{\!|A_2/[\theta]\,P_2|\!\}_s$. $\{\!|A_1/[\theta]\,P_1|\!\}_w = (\Delta_1, \mu_1)$ with $\Delta_1 \subset \Delta$. By induction $|A_1| \subset \Delta_1$. Then $|Y| = |A_1A_2| = |A_1| \subset \Delta$.
- Case $\{\!|Y/[\theta]\,X|\!\}_w = (|Y||X|, \bot)$ is immediate. □

Lemma 4.3. *If* $\{\!|Y/[\theta]\,X|\!\}_s = (\Delta, \mu)$ *or* $\{\!|Y/[\theta]\,X|\!\}_w = (\Delta, \mu)$ *then* $|X| \subset \Delta$.

Proof. Similar to Lemma 4.2 □

Lemma 4.4. *If X is not a data structure, $X \xrightarrow{R} X'$ with R of name Ω and X' a data structure, then $\Omega \prec |X'|$.*

Proof. By induction on $X \xrightarrow{R} X'$, where X is not a data structure.

- Case $X = [\theta]\,P \to B$ is impossible: X' would not be a data structure.
- Case $X = T_1T_2 \xrightarrow{R} T_1'T_2 = X'$ with $T_1 \xrightarrow{R} T_1'$, where T_1 is not a data structure. Since X' is a data structure, T_1' is a data structure. By induction hypothesis $\Omega \prec |T_1'|$. Moreover $|X'| = |T_1'T_2| = |T_1'|$. Thus $\Omega \prec |X'|$.
- Case $X = T_1T_2 \xrightarrow{R} T_1T_2' = X'$ with T_1 not a data structure can not make X' a data structure.
- Case $X = \alpha : Z \xrightarrow{R} \alpha : Z' = X'$ with $Z \xrightarrow{R} Z'$. Z is not a data structure, but Z' is. By induction hypothesis $\Omega \prec |Z'|$. But $|X'| = |\alpha : Z'| = \alpha|Z'|$. Hence $\Omega \prec |X'|$.
- Case $X = R = \alpha : ((\Gamma \cdot [\theta]\,P \to B)A) \xrightarrow{R} X'$. The result has the form $X' = \ulcorner\Omega\urcorner : Z'$. Thus $|X'| = \ulcorner\Omega\urcorner|Z'|$ and $\Omega \prec |X'|$. □

Corollary 4.5. *If X is not a matchable form, $X \xrightarrow{R} X'$ with R of name Ω and X' a matchable form, then $\Omega \prec |X'|$.*

Proof. If X' is data structure, then apply Lemma 4.4. Suppose $X' = \Gamma' \cdot [\theta'] P' \to B'$, which is the only other case. Then $|X'| = \Gamma'$. X is not a matchable form, and can not have the form $\Gamma : x$ which can not be reduced. Thus $X = \Gamma \cdot (T_1 T_2)$. Then $X = \Gamma_0 \alpha : (\Gamma_1 \cdot ([\theta] P \to B) T_2)$ where $R = \alpha : (\Gamma_1 \cdot ([\theta] P \to B) T_2)$. Hence $\Gamma' = \Gamma'_0 \ulcorner \Omega \urcorner \Gamma'_1$ and $\Omega \prec |X'|$. □

Lemma 4.6. *For any terms X, Y with $\{\!|Y/[\theta] X|\!\}_p = \texttt{wait}$, and R a redex of name Ω,*

- *If $Y \xrightarrow{R} Y'$ and $\{\!|Y'/[\theta] X|\!\}_p = (\Delta, \mu)$ then $\Omega \prec \Delta$.*
- *If $X \xrightarrow{R} X'$ and $\{\!|Y/[\theta] X'|\!\}_p = (\Delta, \mu)$ then $\Omega \prec \Delta$.*

Proof. First notice that the base cases for $\{\!|Y/[\theta] X|\!\}_p = \texttt{wait}$ are:

- X is a clipped term but not a matchable form.
- X is a clipped data structure and Y is a clipped term but not a matchable form.

Now by induction on $\{\!|Y/[\theta] X|\!\}_p$.

- Case $\{\!|Y/[\theta] \alpha : Z|\!\}_p = \alpha : \{\!|Y/[\theta] Z|\!\}_p$ with $\{\!|Y/[\theta] Z|\!\}_p = \texttt{wait}$.
  - If $Y \xrightarrow{R} Y'$ with R of name Ω and $\{\!|Y'/[\theta] Z|\!\}_p = (\Delta, \mu)$, by induction $\Omega \prec \Delta$. In this case $\{\!|Y'/[\theta] \alpha : Z|\!\}_p = \alpha : \{\!|Y'/[\theta] Z|\!\}_p = (\alpha\Delta, \mu)$, and $\Gamma \prec \alpha\Delta$ still holds.
  - The case where $Z \xrightarrow{R} Z'$ is similar.
  - Third case is the reduction of $X = \alpha : Z$ at the root: $X \xrightarrow{X} X'$. In this case $X' = \ulcorner \Omega \urcorner : Z'$, and $\{\!|Y/[\theta] X'|\!\}_p = \ulcorner \Omega \urcorner : \{\!|Y/[\theta] Z'|\!\}_p$. Thus $\Delta = \ulcorner \Omega \urcorner \Delta'$ and $\Omega \prec \Delta$.
- Cases $X = \hat{x}$ with $x \in \theta$ and $X = [\tau] P \to B$ do not result in `wait`.
- Case $\{\!|\alpha : Z/[\theta] X|\!\}_p = \alpha : \{\!|Z/[\theta] X|\!\}_p$ is symmetrical of the first case.
- Case $X = Y = \hat{x}$ with $w \notin \theta$ do not result in `wait`.
- Case $\{\!|A_1 A_2/[\theta] P_1 P_2|\!\}_p = \{\!|A_1/[\theta] P_1|\!\}_w \Subset \{\!|A_2/[\theta] P_2|\!\}_s$. There are two subcases here:
  - If $\{\!|A_1/[\theta] P_1|\!\}_w = \texttt{wait}$. Suppose $A_1 \xrightarrow{R} A'_1$ and $\{\!|A'_1/[\theta] P_1|\!\}_w = (\Delta_1, \mu_1)$. By induction $\Omega \prec \Delta_1$. Since $\Delta_1 \subset \Delta$, conclusion $\Omega \prec \Delta$ is immediate. Case $P_1 \xrightarrow{R} P'_1$ is similar.
  - If $\{\!|A_1/[\theta] P_1|\!\}_w = (\Delta_1, \sigma_1)$ and $\{\!|A_2/[\theta] P_2|\!\}_s = \texttt{wait}$, proof is similar.
- Other cases where X and Y are matchable forms do not result in `wait`.
- Base cases, as mentioned above.
  - Case where X is a clipped term but not a matchable form. Reductions of Y do not change anything. Suppose $X \xrightarrow{R} X'$ with X' a matchable form. By Lemma 4.5, $\Omega \prec |X'|$. By Lemma 4.3 $|X'| \subset \Delta$, and thus $\Omega \prec \Delta$.

– Case where X is a clipped data structure and Y is a clipped term but not a matchable form. Reductions of X do not change anything. If Y is a variable, it can not be reduced. Suppose $Y = A_1 A_2$. Reductions in A_2 can not change Y into a matchable form. Then suppose $A_1 \xrightarrow{R} A_1'$, with A_1' a data structure and $Y' = A_1' A_2$. By Lemma 4.4, $\Omega \prec |A_1'|$. By Lemma 4.2, $|Y'| \subset \Delta$. Moreover $|A_1'| = |A_1'|A_2 = |Y'|$, and thus $\Omega \prec \Delta$. □

Lemma 4.7. *For any $X \xrightarrow{R} X'$ with redex R of name Ω, if α is a non initial label created in X' (created occurrence, or residual of a different label), then $\Omega \prec \alpha$.*

Proof. Created occurrences of labels fall in two cases:

- Label $\ulcorner\Omega\urcorner$ at the root of the reduced redex. Then $\Omega \prec \ulcorner\Omega\urcorner$.
- Distinguished labels of the term $\bot$. They are initial and thus not concerned here.

A transformed label can only be of the form $[\Omega, \beta]$, with $\Omega \prec [\Omega, \beta]$. □

4.0.2 Preservation of Sharing Theorem

A term T is said to enjoy the **maximality property** $\mathbb{M}$ (noted $\mathbb{M}(T)$) if for any subredex R of name Ω and subterm $\alpha : Z$ in T, $\Omega \not\prec^+ \alpha$.

Lemma 4.8. *If $\mathbb{S}(T)$, $\mathbb{M}(T)$ and $T \Rrightarrow T'$, then $\mathbb{S}(T')$ and $\mathbb{M}(T')$.*

Proof. $T \Rrightarrow T'$ is by development of all redexes of name Ω_0.
Verification of $\mathbb{M}(T')$: suppose there is a redex R' with name Ω' and a subterm $\alpha' : Z'$ in T', such that $\Omega' \prec^+ \alpha'$.

- If R' is a residual of a redex R of T with name Ω, by Lemma 3.10 $\Omega = \Omega'$. Since $\mathbb{M}(T)$ holds, α can not appear in T, and is created by reduction. For instance $\alpha = \ulcorner\Omega_0\urcorner$. Then $\Omega \prec^+ \ulcorner\Omega_0\urcorner$, with two possible cases:
  - $\Omega = \Omega_0$. But in this case R should have been contracted by the development of redexes of name Ω_0.
  - $\Omega \prec^+ \Omega_0 = \omega_0^1 ... \omega_0^n$. There is a i in $\{1...n\}$ such that $\Omega \prec^+ \omega_0^i$. Let R_0 be a redex of T with name Ω_0. There exists a subterm of R_0 of the form $\omega_0^i : X_0^i$. Then $\Omega \prec^+ \omega_0^i$ breaks $\mathbb{M}(T)$.

  The only other case for α is $\alpha = [\Omega_0, \alpha_0]$ and is similar.
- If R' is created by the reduction, then by Lemma 3.3 $\Omega_0 \prec \Omega \prec^+ \alpha$, and in particular $\Omega_0 \prec^+ \alpha$. Since $\mathbb{M}(T)$ holds, α can not appear in T but is created by the reduction. For instance $\alpha = \ulcorner\Omega_0\urcorner$, then $\Omega_0 \prec \Omega \prec^+ \ulcorner\Omega_0\urcorner$. In particular $\Omega \prec^+ \Omega_0$ or $\Omega = \Omega_0$. In any case, $\Omega_0 \prec^+ \Omega_0$, which is impossible. The only other case $\alpha = [\Omega_0, \alpha_0]$ is similar.

Verification of $\mathbb{S}(T')$: let $\alpha : X'$ and $\alpha : Y'$ be subterms of T'. By case on the origin of both labels α.

- If one is created but the other is a residual of itself:
  - α is created in the development of Ω_0, then by Lemma 4.7 $\Omega_0 \prec \alpha$.
  - α coexists in T with redexes of name Ω_0, thus $\Omega_0 \not\prec^+ \alpha$ by $\mathbb{M}(T)$, and in particular $\Omega_0 \not\prec \alpha$.

The two conclusions are in contradiction !

- If both are created, case on the creations:
  - The label $\ulcorner\Omega_0\urcorner$ is created only at the root of the contractum R'_0 of a redex R_0 of name Ω_0: if $\alpha = \ulcorner\Omega_0\urcorner$ then $\alpha : X' = \alpha : Y' = R'_0$ and in particular $X' = Y'$.
  - The label $[\Omega_0, \alpha_0]$ is created in a function body during substitution, and replaces a label α_0. $\alpha : X'$ and $\alpha : Y'$ are residuals of subterms $\alpha_0 : X$ and $\alpha : Y$ of T. By $\mathbb{S}(T)$ they are affected by the same substitution, and still by $\mathbb{S}(T)$ $X = Y$. Thus $X' = Y'$.
- If both are residuals of themselves: there are subterms $\alpha : X$ and $\alpha : Y$ in T which are ancestors of $\alpha : X'$ and $\alpha : Y'$, with $X = Y$ by $\mathbb{S}(T)$. Cases on the residuals:
  - If $\alpha : X$ and $\alpha : Y$ are disjoint from developped redexes, they are equals to their residuals $\alpha : X'$ and $\alpha : Y'$. In this case $Y' = Y = X = X'$.
  - If $\alpha : X$ is affected by an external substitution: $R_0 = \alpha_0 : ((\Gamma_0 \cdot [\theta_0]\, P_0 \to B_0)A_0)$ with $\alpha : X$ subterm of B_0. If $\theta_0 \# X$, $\alpha : X' = \alpha : X$, which is the same as the previous case. If $\theta_0 \cap fv(X) \neq \emptyset$, the residual of α is $[\Omega_0, \alpha]$, which is not the case. Case for $\alpha : Y$ symmetrical.
  - If $\alpha : X$ is a subterm of A_0 which is in the codomain of σ_0, its residuals are all equal to $\alpha : X$ and as previously $X' = X$. Same holds for $\alpha :$.
  - If $\alpha : X$ contains one or more redex with name Ω_0. If $\alpha : X$ is itself such a redex, it is reduced by the development and has no residual. Suppose these redexes are strict subterms: they are in particular subterms of X. Thus $X \Rrightarrow X'$. Similarly $Y \Rrightarrow Y'$ and by $\mathbb{S}(T)$ $X = Y$. Hence by uniqueness of the complete development, $X' = Y'$. □

Theorem 4.9 (Preservation of Sharing Theorem)**.** *Let $T \Rrightarrow^* T'$ with T an initial term. Then T' enjoys the sharing property.*

Proof. The initial term T enjoys the sharing property and the maximality property. A straightforward induction on the length of $T \Rrightarrow^* T'$ using Lemma 4.8 shows that T' also enjoys these two properties. □

Outcome of this part is a graph implementation of weak pure pattern calculus featuring optimal sharing, which corresponds to fully lazy sharing for *PPC*: when a function body is instantiated, the only modified labels (which means the only copied nodes!) correspond to the substitution slice.

5 The Result of Optimality

Now that an implementation model is defined, this section characterizes a family of strategies over graphs (resp. labelled terms) which always normalize in a minimal number of reduction steps (resp. developments of names). A straightforward corollary will be that these strategies are correct: whenever a normal form exists (unique, by Confluence Theorem 3.12), they reach it (with a minimal number of steps). The first part is informal and puts the focus on the following message: all the tough work toward an optimality result has already been done in previous sections. A formal statement comes next.

A redex R in a term T is said to be **needed** when any reduction $\overrightarrow{\rho} : T \to T'$ to a normal form T' contracts R or at least one of its residuals. A needed strategy reduces only needed redexes.

An axiomatic framework making needed strategies optimal is given in [GK96]. Ingredients are: a notion of residual, a family relation, a contribution relation over families (families are the equivalence classes of the family relation), and a set of terms considered as *results*, each satisfying its own group of axioms. This kind of result is mostly inapplicable if one doesn't know how to find such abstract notions in the concrete system, but the labels of *LWPPC* provide a solution:

- The extant notion of residual is used (Section 2).
- Two redexes are defined to be in the same family if and only if they have the same name (hence each family is assimilated to a name).
- Define the abstract contribution relation as the extant direct contribution relation on labels (Section 3.2).

To form a *Deterministic Family Structure* (that is the name of the abstract concept), these definitions have to satisfy finite development properties (Finite Development Theorem 3.32), finite family developments (deduced from Termination Theorem 3.24), and properties relating contribution relation to creation of redexes (deduced from Redex Stability Lemma 3.10 and a converse property detailed below).

5.0.1 Formal Account of Determinisic Family Structures

A *Deterministic Residual Structure* [GK96] is a rewriting system equiped with a residual relation satisfying the following properties:

- **[FD]** *All developments are terminating; all co-initial complete developments of the same set of redexes end at the same term; and residuals of a redex under all complete co-initial developments of a set of redexes are the same.*
- **[Acyclicity]** *Let r and r_e be two distinct redexes of a term t such that r erases r_e. Then r_e does not erase r.* (r **erases** r_e means that r_e has no residual after the reduction of r)

Lemma 5.1. LWPPC *equiped with its residual relation is a* Deterministic Residual Structure

Proof. Axiom **FD** is the Finite Development Theorem 3.32. **Acyclicity** is satisfied with the following remark: in *LWPPC* if a redex r erases a redex r_e, then r_e is a strict subterm of r. The subterm relation being acyclic, there is no possible cross-erasure. □

Families are linked to history. They are defined on redexes with a full recording of all past reduction in the whole term (that means the whole sequence of reduction). The **redex with history** $\overrightarrow{\rho}; R$ is the redex R at the end of the sequence of reduction $\overrightarrow{\rho}$. The notion of family is formalized by an equivalence relation $\simeq$ on redexes with history, which relates only **coinitial** histories (with same source term). Its equivalence classes are called **families** and noted $Fam(\overrightarrow{\rho}; R)$.

An abstract notion $\simeq_z$ of **zig-zag** which represents minimal requirements for the family relation $\simeq$ is defined as follows: let $\overrightarrow{\rho_1}; R_1$ and $\overrightarrow{\rho_2}; R_2$ be two coinitial redexes with history. Suppose $\overrightarrow{\rho_1}$ has no residual after $\overrightarrow{\rho_2}$ (which means that any *task* of $\overrightarrow{\rho_1}$ is fulfilled or erased by $\overrightarrow{\rho_2}$). Let $\overrightarrow{\rho_2}/\overrightarrow{\rho_1}$ be a complete development of the residuals of $\overrightarrow{\rho_2}$ after $\overrightarrow{\rho_1}$. If R_2 is a residual of R_1 after $\overrightarrow{\rho_2}/\overrightarrow{\rho_1}$, then $\overrightarrow{\rho_1}; R_1 \rhd \overrightarrow{\rho_2}; R_2$ (R_2 is a **copy** of R_1). The relation $\simeq_z$ is the least equivalence relation containing $\rhd$.

A **Deterministic Family Structure** [GK96] is a Deterministic Residual Structure equiped with a family relation $\simeq$ over redexes with history (whose equivalence classes are called families) and a contribution relation $\hookrightarrow$ over families such that the following axioms are satisfied:

- **[Initial]** *For any R_1, R_2 distinct redexes, $Fam(\emptyset; R_1) \neq Fam(\emptyset; R_2)$.*
- **[Zig-zag]** $\simeq_z \subseteq \simeq$
- **[FFD]** *Any reduction sequence that contracts redexes of a finite number of families is finite.*
- **[Creation]** *If $\mathcal{F} \hookrightarrow Fam(\overrightarrow{\rho}; R)$ then $\overrightarrow{\rho}$ contracts at least one redex in $\mathcal{F}$.*
- **[Contribution]** *If after a sequence $\overrightarrow{\rho_p}$ the reduction ρ of a redex R creates a redex R_c, then $Fam(\overrightarrow{\rho_p}; R) \hookrightarrow Fam(\overrightarrow{\rho_p}\rho; R_c)$.*

5.0.2 Correctness & Optimality Corollary

Lemma 5.2. *Suppose M is a matchable form, with $|M| = \alpha_1...\alpha_n$. Then for any $i \in \{1...n\}$, M has a subterm of the form $\alpha_i : Z$.*

Proof. Straightforward induction on the definition of $|M|$. □

Lemma 5.3. *Suppose $\{\!\{A/[\theta]\, P\}\!\}_p = (\delta_1...\delta_n, \mu)$. Then for any $i \in \{1...n\}$, A or P has a subterm of the form $\delta_i : Z$.*

Proof. Straightforward induction on the definition of $\{\!\{A/[\theta]\, P\}\!\}_p$, with Lemma 5.2. □

Lemma 5.4. *Let R be a redex of name $\Omega = \omega_1...\omega_n$. For any $i \in \{1...n\}$, R has a subterm of the form $\omega_i : Z$.*

Proof. Note $R = \alpha : (\Gamma \cdot ([\theta]\, P \to B)A)$ with $\{A/[\theta]\, P\} = (\Delta, \mu)$. Then $\Omega = \alpha\Gamma\Delta$. If ω_i is α or is in Γ, the result is immediate. If ω_i is in Δ then by Lemma 5.3 A or P has a subterm of the form $\omega_i : Z$. □

Lemma 5.5 (Direct Contribution Converse). *Let T be an initial term. Suppose $\overrightarrow{\rho} : T \to T'$, with R' a redex of T' with name Ω'. Suppose there exists a redex of name Ω satisfying $\Omega \prec \Omega'$. Then $\overrightarrow{\rho}$ contracts at least one redex R with name Ω.*

Proof. Since $\Omega \prec \Omega'$, $\Omega' = \omega'_1...\omega'_n$ and there is a i such that $\omega'_i = \ulcorner\Omega\urcorner$ or $\omega'_i = [\Omega, \alpha]$. Hence by Lemma 3.6 a label ω'_i can be created only by reduction of a redex of name Ω. By Lemma 5.4, the redex R' has a subterm of the form $\omega'_i : Z$ and since T is an initial term this label ω'_i has effectively been created by $\overrightarrow{\rho}$. Thus $\overrightarrow{\rho}$ contracts a redex of name Ω. □

Define $\simeq$ as: $\overrightarrow{\rho_1}; R_1 \simeq \overrightarrow{\rho_2}; R_2$ if and only if the redexes R_1 and R_2 have the same name. This is obviously an equivalence relation, whose equivalence classes (the families) are in bijection with redex names. Then the direct contribution relation on labels $\prec$ extends to families.

Theorem 5.6. LWPPC *equipped with $\simeq$ and $\prec$ is a Deterministic Family Structure.*

Proof.

- In an initial source term all redexes have different names, and hence are in different families.
- Suppose $\overrightarrow{\rho_1}; R_1 \rhd \overrightarrow{\rho_2}; R_2$. In particular R_2 is a residual of R_1 after some reduction. Hence by Redex Stability Lemma 3.10 they have the same name, and thus $\overrightarrow{\rho_1}; R_1 \simeq \overrightarrow{\rho_2}; R_2$. Since $\simeq$ is an equivalence relation containing $\rhd$, by definition of $\simeq_z$ axiom **FFD** is satisfied.
- Let $\mathcal{F}$ be a finite set of families. Define $\mathcal{N}$ as the (finite) set of names of families of $\mathcal{F}$. By Termination Theorem 3.24 $\to_{\mathcal{N}}$ is terminating. Hence axiom **FFD** holds.

- Suppose $T \xrightarrow{\overrightarrow{\rho}} T' \xrightarrow{\rho_0:R_0} T''$ with ρ_0 creating R_c in T'', let $\overrightarrow{\rho_1}; R_1$ be a redex with history in $Fam(\overrightarrow{\rho}\rho_0; R_c)$. Note Ω_0 and Ω_1 the names of redexes R_0 and R_1. By definition of families, R_c has also the name Ω_1, and thus by Direct Contribution Lemma 4.1, $\Omega_0 \prec \Omega_1$. Hence by Lemma 5.5 $\overrightarrow{\rho_1}$ has the form $\overrightarrow{\rho_1^1}\rho\overrightarrow{\rho_1^2}$ where ρ contracts a redex R of name Ω_0, with by definition $\overrightarrow{\rho_1^1}; R \in Fam(\overrightarrow{\rho}\rho_0; R_0)$. Hence the **Contribution** axiom is satisfied.
- **Creation** axiom follows immediately Direct Contribution Lemma 4.1. □

Finally, the set of normal forms is an easy example of stable set of results, and hence results of [GK96] apply.

Corollary 5.7. Correctness & Optimality. *Let T be an initial normalizable term. Then any needed reduction of $\Rrightarrow$ reaches the (unique) normal form with a minimal number of steps.*

6 Related Works

The approach used in this paper to derive a graph implementation owes a lot to Blanc, Lévy and Maranget, who described a labelled weak λ-calculus enjoying the sharing property [BLM07]. Their work is generalized here in several ways. First the method used to derive a labelling system is guided by the notion of contribution. This approach enables a slight simplification of their labels (only two syntactic constructs versus three, and also less indirections). Secondly the approach is extended to get results on strategies and not only on the representation of programs. Last but not least, the study is done on a pattern calculus which is a strict generalization of λ-calculus.

Labelled terms representing graphs and results of correctness and optimality are studied by Maranget in [Mar92] for orthogonal first-order term rewriting systems, and even applied to pattern matching *à la ML* through compilation into supercombinators. However, this compilation scheme makes use of the static nature of patterns in Maranget's framework, and hence can not be applied to dynamic patterns. Moreover, such compilation treats pattern matching *a priori* and prevents the direct study of its history in a higher-order setting. Last, please notice that the labelling systems in [Mar92] implement only the sharing of lazy evaluation, and that something new is needed for fully lazy evaluation, as suggested by [BLM07]. Indeed full laziness is closely related to the particular weak reduction used here, which asks for a particular treatment of the substitution slice of each contractum. This can be done either by using as many rules as there are possible substitution slices (that means infinitely many), or by redefining a labelled substitution (which is the solution used here).

A lot of results on confluence, developments, descendants and origin tracking also exist in general (higher-order) rewriting frameworks [Mar92, Ter03], and an encoding of *PPC* into *Combinatory Reduction Systems (CRS)* in particular is suggested by Klop, van Oostrom and de Vrijer in [KvOdV08]. Unfortunately this encoding has one major drawback from the implementational point of view. Indeed it is based on a rule scheme that generates one rule for each term acceptable as a pattern (that is for successful matchings) and has to be extended with more complex schemes for matching failures. This enumeration of all possible matchings leads to a *CRS* with infinetely many rules which can be useful for understanding key notions of origins [Ter03] and providing immediate proofs of some results (such as confluence [vOvR94] or finite family developments [Bru08]), but can not be used as such for implementation. Moreover, remark that infinitely many rules yield an infinite alphabet of rule names for labelling, whereas in *LWPPC* all is built with one rule and the finite set of labels that decorates the original term.

More generally, a term (or a higher-order) rewriting system asks for an infinite enumeration of rules to model matching against all possible patterns. Each rule has a fixed shape which

determines the relevant parts of the argument and the way labels are handled. On the other hand, pattern matching calculi such as *PPC* use for all patterns a unique matching algorithm which identifies dynamically the parts of the pattern and of the argument that are relevant for matching. This leads to a new view on labelling which is addressed in this paper.

The optimality result could also have been proved in a more standard way [Mar91, Yos94], namely by stating a one-step diamond property for the development of names in *LWPPC* (which can be proved using the same central result: Finite Developments Theorem 3.32). The abstract approach by Glauert and Khasidashvili [GK96] is preferred here for its modularity. It has already been successfully applied on wide classes of higher-order rewriting systems, and could work on an encoding of *PPC* by using general results of [Ter03, Bru08]. An alternative method is proposed here, which is a direct reuse of the general results on *LWPPC* (and hence is more self-contained).

Also notice that *PPC* has already an implementation, known as the programming language Bondi developed by Jay [Bon] [Jay09, Part 3]. The graph implementation presented here could help improving the efficiency of the Bondi evaluator by introducing more sharing.

7 Conclusion and Prospects

Enriched pattern matching paradigms allowing dynamic patterns, such as *PPC*, improve expressive power and usability of functional programming languages by offering path and pattern polymorphisms to the programmer. Unfortunately they also invalidate usual optimizations performed by compilers on functions defined by cases, which is a severe drawback when it comes to implementation.

This paper lays the foundations of a sharing theory for these frameworks, with the goal of overcoming this difficulty. Originality of this work is in the way a careful analysis of the contribution relation between redexes can be used to derive a graph implementation as well as optimal reduction strategies. This work is also the first application of such an analysis to a pattern matching framework based on a concise definition of matching instead of an enumeration of all possible matchings. Difficulties specific to this feature such as non-local contributions and the handling of failures are tackled.

The first result is a graph implementation featuring fully lazy sharing (Preservation of Sharing Theorem 4.9), which combines a fairly good level of sharing with the possibility of an efficient implementation. This result is associated with a description of reduction strategies that are correct and efficient (Correctness & Optimality Corollary 5.7).

However, this paper is just a first step toward a more ambitious program: as mentioned in the introduction the aim of this work is to provide sharing mechanisms that share pattern matching steps. This kind of sharing is limited in the current graph implementation, due to the implicit treatment of pattern matching: in *PPC* each matching is performed globally as an atomic operation, and can be shared only as a whole. This hides the fact that any matching is composed of several elementary matching steps which could be shared individually.

In order to get this finer control, the technology presented in this paper has to be extended to the *Explicit Pure Pattern Calculus* [Bal08]: a variant of *PPC* with explicit pattern matching, where all matching steps are visible. In this extended framework every single piece of a pattern could be shared independently of the other parts, and if two patterns in the same function share some structure, then corresponding pattern matching steps could also be shared! Furthermore, to ensure that sharing is not lost along sequences of pattern matching cases, an explicit *alternative case* operator has to be added to the calculus (as it is done in the *Extension Calculus* [Jay09]). A third point that has to be addressed is the analysis of the consequences of the requirement of a deterministic strategy for pattern matching operations. Few strategies

are available for now, but explicit matching will enrich them.

This allows to hope for a functional programming language which would feature dynamic patterns and still have an efficient implementation: the sharing model could be used directly to manage pointers and copies in an efficient graph implementation such as [SW04], or to guide the design of an advanced implementation by interaction nets in the style of [Mac04, FMSW09].

The *explicit matching* framework will enrich also the optimality result. Firstly, an optimal strategy taking into account every single pattern matching operation is a way to manage all the low-level tests induced by functions defined by multiple cases. Secondly, whereas *PPC* only allows an abstract definition of needed strategies, some of them can be *effectively* described as variants of *leftmost-outermost* in the explicit framework.

8 Acknowledgments

Special thanks to my advisor Delia Kesner. Many thanks also to Luc Maranget, Zurab Khasidashvili, Maribel Fernández and Barry Jay for comments and suggestions on this work, and feedback on my project.

References

[AG98] A. Asperti and S. Guerrini. *The optimal implementation of functional programming languages*. Cambridge University Press, 1998.

[Bal08] T. Balabonski. Calculs avec motifs dynamiques, Master's thesis, Université Paris Diderot, 2008. Available as `http://www.pps.jussieu.fr/~balabons/Recherche/Balabonski_Rapport_M2.pdf`.

[BLM07] T. Blanc, J.-J. Lévy, and L. Maranget. Sharing in the Weak Lambda-Calculus Revisited. In *Reflections on Type Theory, Lambda Calculus and the Mind* Essays Dedicated to Henk Barendregt on the Occasion of his 60th Birthday, December 2007.

[Bon] Bondi programming language. `http://bondi.it.uts.edu.au/`.

[Bru08] S. Bruggink. *Equivalence of Reductions in Higher-Order Rewriting*. Ph.D. thesis, 2008.

[ÇH98] N. Çağman and J. R. Hindley. Combinatory weak reduction in lambda calculus. *Theor. Comput. Sci.*, 198(1-2):239–247, 1998.

[Cir00] H. Cirstea. *Rewriting Calculus: Foundations and Applications*. Ph.D. thesis, 2000.

[DLLL05] D. Dougherty, P. Lescanne, L. Liquori, and F. Lang. Addressed Term Rewriting Systems: Syntax, Semantics, and Pragmatics: Extended Abstract. *ENTCS*, 127(5):57–82, 2005.

[FM01] F. Le Fessant and L. Maranget. Optimizing pattern matching. In *ICFP*, pages 26–37, 2001.

[FMSW09] M. Fernández, I. Mackie, S. Sato, and M. Walker. Recursive functions with pattern matching in interaction nets. *ENTCS*, 253(4):55–71, 2009.

[GK96] J. Glauert and Z. Khasidashvili. Relative Normalization in Deterministic Residual Structures. In *CAAP*, pages 180–195, 1996.

[HG91] C.K. Holst and D.K. Gomard. Partial evaluation is fuller laziness. *SIGPLAN Not.*, 26(9):223–233, 1991.

[HJ03] R. Hinze and J. Jeuring. Generic haskell: Practice and theory. In *Generic Programming*, volume 2793 of *LNCS*, pages 1–56, 2003.

[Jay04] B. Jay. The pattern calculus. In *TOPLAS*, volume 26(6), pages 911–937, 2004.

[Jay09] B. Jay. *Pattern Calculus: Computing with Functions and Data Structures*. Springer, 2009.

[JK06] B. Jay and D. Kesner. Pure pattern calculus. In *ESOP, LNCS 3942*, pages 100–114, 2006.

[JK08] B. Jay and D. Kesner. Patterns as first-class citizens. Technical report, 2008. Available as `http://hal.archives-ouvertes.fr/hal-00229331/fr/`.

[JK09] B. Jay and D. Kesner. First-class patterns. *J. Funct. Programming*, 19(2):191–225, 2009.

[Jon87] S. Peyton Jones. *The Implementation of Functional Programming Languages*. Prentice-Hall, Inc., 1987.

[KvOdV08] J. W. Klop, V. van Oostrom, and R. de Vrijer. Lambda calculus with patterns. *TCS*, 398:16–31, 2008.

[Lé78] J.-J. Lévy. *Réductions correctes et optimales dans le lambda-calcul*. Ph.D. thesis, 1978.

[Lé80] J.-J. Lévy. Optimal reductions in the lambda-calculus. In *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalisms*, pages 159–191, 1980.

[Mac04] I. Mackie. Efficient lambda-evaluation with interaction nets. In *RTA*, pages 155–169, 2004.

[Mar91] L. Maranget. Optimal Derivations in Weak Lambda-calculi and in Orthogonal Terms Rewriting Systems. In *POPL*, pages 255–269, 1991.

[Mar92] L. Maranget. *La stratégie paresseuse*. Ph.D. thesis, 1992.

[Mar08] L. Maranget. Compiling pattern matching to good decision trees. In *ML*, pages 35–46, 2008.

[Sin08] F.-R. Sinot. Complete laziness: a natural semantics. *ENTCS*, 204:129–145, 2008.

[SW04] O. Shivers and M. Wand. Bottom-up β-reduction: Uplinks and λ-DAGs. Technical Report RS-04-38, BRICS, December 2004.

[Ter03] Terese. *Term Rewriting Systems*. Cambridge University Press, 2003.

[vOvR94] V. van Oostrom and F. van Raamsdonk. Weak orthogonality implies confluence: the higher-order case. In *LFCS'94*, pages 379–392, 1994.

[Wad71] C. P. Wadsworth. *Semantics and Pragmatics of the Lambda Calculus*. Ph.D. thesis, 1971.

[Yos94] N. Yoshida. Optimal reduction in weak-λ-calculus with shared environments. *Journal of Computer Software*, 11(5):2–20, September 1994.