# Optimal Derivations in Weak Lambda-calculi and in Orthogonal Terms Rewriting Systems.

Luc Maranget *

## Abstract

We introduce the new framework of Labeled Terms Rewriting Systems ($\mathrm{T}^l\mathrm{RS}$), a general framework to express sharing in Term Rewriting Systems (TRS). For Orthogonal $\mathrm{T}^l\mathrm{RS}$, an important subclass of $\mathrm{T}^l\mathrm{RS}$, we characterize optimal derivations. This result is applied to weak $\lambda$-calculi, showing the optimality of the *lazy* strategy, that is, the call-by-name with sharing strategy. The result is also valid in the presence of $\delta$-rules, as in PCF. Orthogonal $\mathrm{T}^l\mathrm{RS}$ is also useful as a calculus for proving syntactic properties of functional languages.

## 1 Compilation of the $\lambda$-calculus

Most compilers for functional languages translate their source language into some enriched $\lambda$-calculus [17], and then, compile this intermediate language to a low-level language, such as mutually recursive supercombinators, as in LML [2, 10], or categorical combinators, as in CAML [4]. These low-level languages define different forms of weak $\beta$-reduction. We now describe two of these low-level languages, supercombinators and explicit substitutions, and gradually introduce our extensions to the TRS formalism.

### 1.1 Compilation to supercombinators

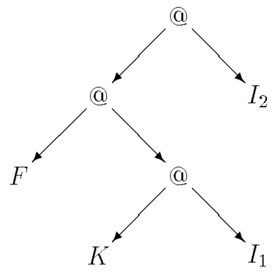
Consider the following $\lambda$-expression :

$$(\lambda f.f\ (\lambda x.f\ x))\ ((\lambda x.\lambda y.x)\ (\lambda x.x))\ (\lambda x.x)$$

The compilation into supercombinators yields a set of combinator definitions and an expression to be computed :

$$\boxed{\begin{array}{rcl} I_1\ x & = & x \\ I_2\ x & = & x \\ K\ x\ y & = & x \\ F f x & = & f\ (G\ f\ x) \\ G\ f\ x & = & f\ x \\ \hline \multicolumn{3}{c}{F\ (K\ I_1)\ I_2} \end{array}}$$

Note how the variable $f$, free in $(\lambda x.f\ x)$, is turned into a locally bound variable which is supplied while calling the supercombinators $F$ and $G$. This process is known as $\lambda$-lifting [2]. These expressions are graphs, to be reduced by the method of lazy graph reduction [2, 17]. There is an implicit binary application node, written @ in the following. As an example, the expression $F\ (K\ I_1)\ I_2$ is equivalent to $@(@(F,\ @(K,\ I_1)),\ I_2)$, or graphically :




Substituting "$\rightarrow$" for "$=$" in the supercombinators definitions yields the rules $\{I_1 x \rightarrow x, \ldots\}$, of the general form $\alpha_i \rightarrow \beta_i$ , to produce a classical Term Rewriting System. The elementary one-step reduction relation, written $M \rightarrow M'$, is induced by the following axioms and inference rules :

$$\text{(Red)}\quad \sigma(\alpha_i) \rightarrow \sigma(\beta_i)$$

$$\text{(App1)}\quad \frac{M \rightarrow M'}{@(M,\ N) \rightarrow @(M',\ N)}$$

$$\text{(App2)}\quad \frac{N \rightarrow N'}{@(M,\ N) \rightarrow @(M,\ N')}$$

*INRIA Rocquencourt, BP 105, 78153 LE CHESNAY CEDEX, FRANCE.

In the rules above, $\sigma$ stands for any substitution, that is, a function mapping terms to terms, such that, $\sigma(f(M_1, M_2, \ldots M_n)) = f(\sigma(M_1), \sigma(M_2), \ldots \sigma(M_n))$. This system is left-linear (no variable appears twice in the left-hand sides of the rules) and has no critical pairs (the left-hand sides of the rules do not overlap with a subterm of another left-hand side). It is an *orthogonal* system in the sense of [9].

## 1.2 Compilation to weak $\lambda$-calculus with explicit substitution

There is no simple definition of weak reduction in the traditional $\lambda$-calculus. If reduction under the $\lambda$'s is not allowed (i.e. if the $(\xi)$ rule is eliminated), the Church-Rosser property of the $\lambda$-calculus no longer holds :

$$\begin{array}{c} \lambda y.y\ ((\lambda x.x)\ (\lambda x.x)) \\ \uparrow \\ (\lambda xy.y\ x)\ ((\lambda x.x)\ (\lambda x.x)) \\ \downarrow \\ (\lambda xy.y\ x)\ (\lambda x.x) \\ \downarrow \\ \lambda y.y\ (\lambda x.x) \end{array}$$

Although some deterministic reduction strategies in the $\lambda$-calculus may be qualified as "weak strategies", as long as they prohibit reduction under the $\lambda$'s, there is no straightforward way to define and study such strategies. A general approach is to map $\lambda$-terms onto a new formal system. In this new framework, it is natural to define a reduction resembling "$\beta$-reduction with no reduction under the $\lambda$'s", and which possesses the Church-Rosser property, inducing a kind of weak-calculus over $\lambda$-terms. This can be done with supercombinators, as introduced in the previous section, or with a fixed set of combinators, as in [7], where combinatory logic and weak combinatory equality are proposed as a basis for defining "weak $\lambda$-equality". These solutions are a bit contrived, since they involve a non-trivial translation process from the $\lambda$-calculus into a (somewhat) different framework, $\lambda$-lifting in the case of supercombinators, and the so called "abstraction algorithm" in the case of combinatory logic.

A simpler approach is compilation to a restricted version of the $\lambda\sigma$ calculus, one of the many $\lambda$-calculi with explicit substitution as introduced in [1, 6]. These calculi are similar, and very close to the original $\lambda$-calculus. Terms are defined by :

$$\begin{array}{ll} \lambda\text{-terms} & M ::= \mathtt{n} \mid M\ N \mid \lambda M \mid M[s] \mid X \\ \text{substitutions} & s ::= M.s \mid id \mid x \end{array}$$

where $\mathtt{n}$ is any number greater or equal than $\mathtt{1}$ and $X$, $x$ are free variables representing unknown terms and substitutions. The substitution $id$ is the identity substitution. We use the De Bruijn's notation, with variables names replaced by indices $\mathtt{n}$ referring to the $n^{\underline{th}}$ surrounding $\lambda$ binder. As an example, the $\lambda$-terms $\lambda x.x$ and $\lambda x.\lambda y.x$ become $\lambda\mathtt{1}$ and $\lambda\lambda\mathtt{2}$ in De Bruijn's notation. This notation is well-suited to formal systems, eliminating the $\alpha$-conversion step needed for the $\lambda$-calculus with named variables.

The compilation process is just a translation of the original $\lambda$-term into De Bruijn notation, and the addition of the identity environment :

$$\begin{array}{c} (\lambda f.f\ (\lambda x.f\ x))\ ((\lambda x.\lambda y.x)\ (\lambda x.x))\ (\lambda x.x) \\ \Downarrow \\ ((\lambda\mathtt{1}\ (\lambda\mathtt{2}\ \mathtt{1}))\ ((\lambda\lambda\mathtt{2})\ (\lambda\mathtt{1}))\ (\lambda\mathtt{1}))[id] \end{array}$$

The reduction rules are a subset of the general rules of the full $\lambda\sigma$-calculus :

$$\begin{array}{ll} \text{(WeakBeta)} & \alpha_1 = (\lambda M)[s]\ N \rightarrow M[N.s] = \beta_1 \\ \text{(App)} & \alpha_2 = (M\ N)[s] \rightarrow M[s]\ N[s] = \beta_2 \\ \text{(FVar)} & \alpha_3 = \mathtt{1}[M.s] \rightarrow M = \beta_3 \\ \text{(RVar)} & \alpha_4 = \mathtt{n+1}[M.s] \rightarrow \mathtt{n}[s] = \beta_4 \end{array}$$

The rule (RVar) stands for an infinity of rules (RVar2), (RVar3),... The elementary one-step reduction relation is the relation $M \rightarrow M'$ on terms induced by :

$$\text{(Red)} \quad \sigma(\alpha_i) \rightarrow \sigma(\beta_i)$$

$$\text{(AppSubst2)} \quad \frac{s \rightarrow s'}{M[s] \rightarrow M[s']}$$

$$\text{(App1)} \quad \frac{M \rightarrow M'}{M\ N \rightarrow M'\ N} \qquad \text{(App2)} \quad \frac{N \rightarrow N'}{M\ N \rightarrow M\ N'}$$

$$\text{(Cons1)} \quad \frac{M \rightarrow M'}{M.s \rightarrow M'.s} \qquad \text{(Cons2)} \quad \frac{s \rightarrow s'}{M.s \rightarrow M.s'}$$

Again there are some syntactic shortcuts. This reduction system is called the weak $\lambda\sigma$-calculus, it is fully characterized by the four reduction rules $\{\ldots \alpha_i \rightarrow \beta_i \ldots\}$, and by a *domain* function $dom()$, which maps function symbols to *allowed* directions of reduction. Here we have, $dom(@) = \{1, 2\}$, $dom([]) = \{2\}$, $dom(.) = \{1, 2\}$ and $dom(\lambda) = \emptyset$. In other words, it is not possible to compute under $\lambda$-abstractions or in the non-environment part of a closure, hence the name "weak" $\lambda\sigma$-calculus. These restrictions allow great simplifications to the full $\lambda\sigma$ calculus which has 24 reduction rules. We recall, without further development, that the $\lambda\sigma$ calculus implements the $\lambda$-calculus. See [1, 6] for detailed explanations and [14] for examples of descriptions of abstract machines implementing strategies of weak reduction in the framework of the weak $\lambda\sigma$-calculus.

In the rest of this paper, we shall investigate the structure of the derivations in *conditional* TRS. That is, TRS with a restricting domain function. We now use the example of the weak $\lambda\sigma$-calculus to recall some basic notions on terms and TRS. See [9] for a complete introduction to these notions.

Each subterm in a term is characterized by its *occurrence* which is either $\epsilon$ for the whole term or a sequence of integers $n_1.n_2\ldots n_m$ representing the access path to it. For instance, in $M = (\lambda \mathtt{1})[s]\ \mathtt{2}[s]$, the subterms at occurrences 1.1.1 and 2.2 are $\mathtt{1}$ and $s$, or $M/1.1.1 = \mathtt{1}$ and $M/2.2 = s$ . Let $u$, $v$ and $w$ be occurrences. If $u = vw$, the occurrence $v$ is a *prefix* of $u$ or *above* $u$ ($v \preceq u$ ). If $w \neq \epsilon$, then $v$ is a *proper prefix* of $u$ ($v \prec u$). If neither $u$ is a prefix of $v$ nor $v$ a prefix of $u$ then $u$ and $v$ are said to be *disjoint* ($u \mid v$). If $u$ and $v$ are disjoint, then there exist a maximum common prefix $w$, with $u = w.n_u.u'$, $v = w.n_v.v'$ and $n_u \neq n_v$. If $n_u < n_v$ then $u$ is to the *left* of $v$. The *leftmost-outermost* ordering on occurrences $<_{lo}$ is defined by : $u <_{lo} v$, iff the occurrence $u$ is above $v$ or to the left of $v$. The leftmost-outermost ordering is a total order relation on the set of the valid occurrences of a given term.

Occurrences give a compact way to express reductions. In the proof tree of $A: M \rightarrow N$ there is one and only one use of the (Red) axiom, the occurrence $u$ of the subterm $\sigma(\alpha)$ in $M$ involved is sufficient to characterize the reduction $A$. This is written $A: M \xrightarrow{u} N$. The subterm $\sigma(\alpha)$ is the *redex* contracted by $A$, whereas $\alpha$ is the *name* contracted by $A$. Occurrences are followed across reductions by the notion of *residuals*. As an example, in the reduction :

$$A: (\lambda \mathtt{1})[s]\ \mathtt{2}[s] \xrightarrow{\epsilon} \mathtt{1}[\mathtt{2}[s].s]$$

the only residual of 1.1.1 is 1 and the only residual of 2.2 is 2.1.2. The notation $u\backslash A$ stands for the set of the residuals of an occurrence $u$ by a reduction $A$. The contracted occurrences (i.e. the non-variable ones inside the name $\alpha$ contracted) have no residual. In the example $\epsilon\backslash A = 1\backslash A = 1.1\backslash A = 1.2\backslash A = \emptyset$. There may be more than one residual, in $B: (M\ N)[s] \rightarrow M[s]\ N[s]$, we have $2\backslash B = \{1.2, 2.2\}$. There may be none of them, even if the original occurrence is not contracted, as in $C: \mathtt{1}[M.s] \rightarrow M$, where $2.2\backslash C = \emptyset$. Some occurrences in the result term are not residuals of any initial occurrence, they are *created occurrences*. For instance, the reduction $A$ creates the occurrences $\epsilon$ and 2 in the final term $\mathtt{1}[\mathtt{2}[s].s]$.

In the weak $\lambda\sigma$ calculus, in contrast with conventional TRS, reduction is not allowed everywhere. A distinction between *allowed* and *forbidden* occurrences can be made. A subterm at occurrence $u = n_1.n_2\ldots n_m$ is allowed, iff,

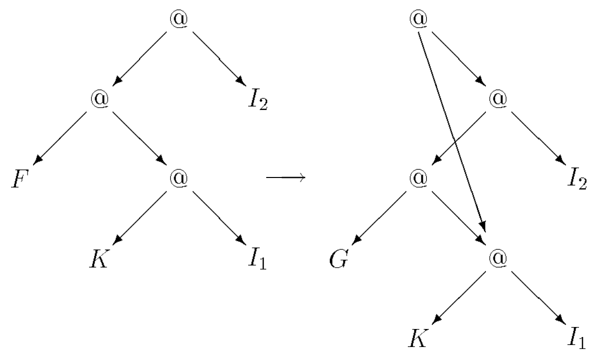



Figure 1: Reduction on graphs

for every $i$, the integer $n_{i+1}$ is in the domain of the function symbol at occurrence $n_1.n_2\ldots n_i$. For instance, in the term $((\lambda M)[s]\ N)[t]$, the allowed occurrences are $\epsilon$, and 2, all the other occurrences are forbidden, because they are prefixed by 1 ($1 \notin dom([\,])$). Therefore, the critical pair between rules (App) and (WeakBeta),

$(\lambda M)[s][t]\ N[t] \leftarrow ((\lambda M)[s]\ N)[t] \rightarrow (M[N.s])[t]$
does not exist, since only the reduction $\leftarrow$ is possible.

Here, a redex in a term $M$ is any subterm of the form $\sigma(\alpha)$ *at an allowed occurrence.* In the weak $\lambda\sigma$-calculus, redexes never overlap. More precisely, let $\sigma_1(\alpha_1)$ and $\sigma_2(\alpha_2)$ be two redexes in a term $M$ at allowed occurrences $u_1$ and $u_2$. Then we never have $u_2 = u_1 v_1$ where $v_1$ is the occurrence of a subterm inside $\alpha_1$, and $\alpha_1/v_1$ is not a variable, except for the trivial case $u_1 = u_2$, $\alpha_1 = \alpha_2$. As in the case of standard TRS, it is sufficient to check the non-overlapping condition on the left-hand sides of the reduction rules only.

In this sense, the weak $\lambda\sigma$ calculus has no critical pairs. As, in addition, it is left linear, we call it an Orthogonal conditional TRS.

# 2 Sharing

The rules with non linear right-hand sides introduce sharing. For instance, in the case of the supercombinators, the reduction of the expression $F\ (K\ I_1)\ I_2$ at occurrence $\epsilon$ gives $(K\ I_1)\ (G\ (K\ I_1)\ I_2)$. An ordinary term cannot express the sharing of the two instances of $(K\ I_1)$. To capture this sharing, we need to reduce graphs, instead of terms, as shown in figure 1, in an informal representation.

The Term Rewriting formalism is unable to express the leftmost-outermost with sharing strategy, or "lazy strategy" of reduction of supercombinators, as it is implemented in the G-machine [2, 10]. It can only express the leftmost-outermost or "call-by-name" strategy.

To express subterm sharing we introduce *labels* in terms. Each subterm $f(M_1, M_2, \ldots M_n)$ is associated with a label $l$. This is written as $f^l(M_1, M_2, \ldots M_n)$ or $\tau(f^l(M_1, M_2, \ldots M_n)) = l$, where $\tau$ is the *top label* function. A label is the "address" of a subterm. Sharing is expressed by equality of labels : a labeled representation of the right graph above is

$$@^{l_1}(@^{l_2}(K^{l_3}, I_1^{l_4}), @^{l_5}(@^{l_6}(G^{l_7}, @^{l_2}(K^{l_3}, I_1^{l_4})), I_2^{l_8})).$$

Not all labeled terms represent graphs : the term $@^l(I_1^m, @^l(I_1^m, x))$ or $(I_1^m(I_1^m(x))^l)^l$ cannot be associated to a graph. We define which terms are graphs :

**Definition 2.1 (DAG)** *Let $M$ be a labeled term in $\mathcal{T}^l$. Then, $M$ is a Directed Acyclic Graph (DAG), iff for all $M_1$ and $M_2$ subterms of $M$, which are not variables, $\tau(M_1) = \tau(M_2)$ implies $M_1 = M_2$.*

As $M_1$ and $M_2$ are labeled terms, this means that, not only the structures of $M_1$ and $M_2$ are equal, but also every address (label) of the corresponding subterms in $M_1$ and $M_2$. This definition of DAGs prohibits cycles. Assume on the contrary that $M_1$ and $M_2$ are two identical subterms of $M$, at occurrences $u_1$ and $u_2$, such that $u_1$ is a proper prefix of $u_2$. Then $M_1 = M_2$ could not be a finite term, since the size (defined in the obvious way) of $M_1$ should be strictly greater than the size of $M_2$.

The interaction of labels with allowed and forbidden occurrences is delicate. It is important not to let reductions change redex-like terms (i.e. terms of the form $\sigma(\alpha)$) from non-redex state to redex state or vice-versa. Otherwise, there could be a situation where some shared subterm is or is not a redex, depending on the instance you are considering. Even if reduction is started from a term with no sharing. Two conditions are sufficient to avoid this problem (see the definition of the residual redexes in section 3.1, as well as lemma 4.3 for more details).

(I) The residuals of allowed occurrences are allowed occurrences.

(II) The residuals of forbidden occurrences are forbidden occurrences.

It is important to notice that these condition are easily decidable, given the domain function and the reduction rules. The set $\mathcal{V}(M)$ of the variables of a term $M$ may be divided into the set $\mathcal{V}_a(M)$ of allowed variables and the set $\mathcal{V}_f(M)$ of forbidden variables, with respect to their occurrences. For any term $M$, we have $\mathcal{V}_a(M) \cup \mathcal{V}_f(M) = \mathcal{V}(M)$. If $M$ is the left-hand side of a reduction rule $\alpha_i \to \beta_i$, then $\mathcal{V}_a(\alpha_i) \cap \mathcal{V}_f(\alpha_i) = \emptyset$, by the linearity of $\alpha_i$. Then, the two conditions

(1) $\mathcal{V}_a(\beta_i) \subseteq \mathcal{V}_a(\alpha_i)$

(2) $\mathcal{V}_f(\beta_i) \subseteq \mathcal{V}_f(\alpha_i)$

on the reduction rules $\alpha_i \to \beta_i$, are sufficient to enforce (I) and (II). For instance, for rule (RVar) in the weak $\lambda\sigma$-calculus :

(1) $\mathcal{V}_a(\mathtt{n}[s]) = \{s\} \subseteq \mathcal{V}_a(\mathtt{n+1}[M.s]) = \{M, s\}$

(2) $\mathcal{V}_f(\mathtt{n}[s]) = \mathcal{V}_f(\mathtt{n+1}[M.s]) = \emptyset$

Whereas for rule (WeakBeta) :

(1) $\mathcal{V}_a((\lambda M)[s]\ N) = \mathcal{V}_a(M[N.s]) = \{N, s\}$

(2) $\mathcal{V}_f((\lambda M)[s]\ N) = \mathcal{V}_f(M[N.s]) = \{M\}$

# 3 Labeled conditional reduction

In this section, we fully define conditional reduction on labeled terms ($\mathrm{T}^l$RS). The labels we introduce are more than just addresses in a graph, they are designed in order to capture the history of the subterm that they decorate. Further in this paper, we shall prove that these sophisticated labeled terms may be abstracted into simpler labeled terms that indeed represent acyclic graphs.

Let $\mathcal{V} = \{x, y, \ldots\}$ be the set of variables. Let $\mathcal{F}_n$ be the set of function symbols of arity $n$ and $\mathcal{F} = \cup\{\mathcal{F}_n \mid n \geq 0\}$ the set of all function symbols. In the case of the weak $\lambda\sigma$ calculus, $\mathcal{F}_1 = \{\lambda\}$ and $\mathcal{F}_2 = \{@, [\,], .\}$. Let $\{a, b, \ldots\}$ be the *atomic labels* from a set of symbols $\mathcal{A}$. The sets $\mathcal{T}^l$ of *labeled terms*, $\mathcal{E}$ of *elementary labels* and $\mathcal{L}$ of *labels* are defined by the following grammar :

$$\begin{array}{rl} \mathcal{T}^l\!: & M ::= x \mid x^l \mid f^l(M_1, M_2, \ldots M_n) \\ \mathcal{E}\!: & e ::= a \mid <M, a> \\ \mathcal{L}\!: & l ::= e \mid l.e \end{array}$$

Labels can be omitted on variables. The operation "." on labels is the concatenation. It is associative. It is convenient to extend the set of labels with the empty label $\epsilon$. Unlabeled variables are then written $x^\epsilon$, and we have $\epsilon.l = l.\epsilon = l$. The *top label* function $\tau\colon \mathcal{T}^l \to \mathcal{L} \cup \{\epsilon\}$ is then defined by :

$$\begin{aligned} \tau(x^l) &= l \\ \tau(f^l(M_1, M_2, \ldots M_n)) &= l \end{aligned}$$

The dot operator is extended to $\mathcal{L} \cup \{\epsilon\} \times \mathcal{T}$ :

$$\begin{aligned} l.x^m &= x^{l.m} \\ l.f^m(M_1, M_2, \ldots M_n) &= f^{l.m}(M_1, M_2, \ldots M_n) \end{aligned}$$

The terms that contribute to the creation of a subterm are introduced into its label with the "catch" operator, $<M, a>$. This operator is also extented to (atomically labeled) terms as a "diffusion" operation :

$$\begin{aligned} <M, x^a> &= x^{<M, a>} \\ <M, f^a(\ldots, N_i, \ldots)> &= f^{<M, a>}(\ldots, <M, N_i>, \ldots) \end{aligned}$$

Substitutions on labeled terms are similar to substitutions on standard terms, with the addition that they also operate on labeled variables :

$$\sigma(x^l) = l.\sigma(x)$$
$$\sigma(f^l(M_1,\, M_2, \ldots M_n)) = f^l(\sigma(M_1),\, \sigma(M_2), \ldots \sigma(M_n))$$

Reduction rules are labeled versions of the rules $\{\ldots,\, \alpha_i \rightarrow \beta_i,\, \ldots\}$ of an unlabeled system, with two important restrictions.

(1) Each $\beta_i$ is labeled, *including variables*, with atomic labels, giving $B_i$.

(2) Each rule $\alpha_i \rightarrow \beta_i$ is replaced by a set of labeled rules, one for every *constrained* labeling $A_i$ of $\alpha_i$. A constrained labeling of $\alpha_i$ is a labeling for which *the top label of $\mathcal{A}_i$ is an elementary label and the variables in $A_i$ are not labeled.* For every constrained labeling $A_i$ of $\alpha_i$, the new rule is $A_i \rightarrow \,<\!A_i,\, B_i\!>$.

Consider, for instance, the symbol sets $\mathcal{F}_1 = \{I\}$, $\mathcal{F}_0 = \{A\}$ and $\mathcal{V} = \{x\}$, the unlabeled rules $I(x) \rightarrow x$ and $A \rightarrow I(A)$. Then, corresponding valid sets of labeled rules may be written as $I^e(x) \rightarrow x^{<I^e(x),\, a>}$ and $A^e \rightarrow I^{<A^e,\, a>}(A^{<A^e,\, b>})$, where $e$ stands for any elementary label, whereas $a$ and $b$ are fixed atomic labels. Valid reduction rules for the labeled weak $\lambda\sigma$-calculus are represented in figure 2. In this figure, the names $\alpha_i$ are here only for the sake of readability ; they should be textually substituted by the full left-hand sides.

Instead of considering only one-step reduction, we opt for a more general notion of reduction, the parallel reduction relation, as introduced by Tait and Martin-Löf. Parallel reduction is defined in figure 3. The inference rules (red) and (rec) represent collections of rules, one for each labeled reduction rule $\alpha \rightarrow \beta$, and one for each non-nullary function symbol $f$. For the rule (red), recall that $\mathcal{V}_a(\alpha)$ and $\mathcal{V}_f(\alpha)$ are the set of allowed and forbidden variables in $\alpha$. For rule (rec), recall that $dom(f)$ is the set of allowed directions from symbol $f$. In the case of the weak $\lambda\sigma$-calculus, the parallel reduction relation is defined by figure 4.

We now give the precise definition of a redex in the context of labeled conditional reduction.

**Definition 3.1 (Redex)** *Let $M$ be a term. Let $R$ be a subterm of $M$, at occurrence $u$. The subterm $R$ is a redex, iff the occurrence $u$ is allowed and $R$ is in the form $l.\sigma(\alpha)$, where $\alpha$ is the left-hand side of a reduction rule, $\sigma$ is a substitution, and $l$ is a (possibly empty) label. The left-hand side $\alpha$ is the* name *of the redex $R$.*

That is, a subterm in $M$, at occurrence $u$ is a redex, iff there exists a reduction starting from $M$, of which proof tree includes the application of a (red) rule, at occurrence $u$.

In the following, we study Orthogonal T$^l$RS only, i.e. the systems defined by a set of labeled reduction rules $\{\ldots,\, \alpha_i \rightarrow \beta_i \ldots\}$ and a domain function, such that :

(1) A variable does not appear twice in the $\alpha_i$'s (left-linearity).

(2) The Names of redexes do not overlap in terms (no critical pairs).

(3) For any reduction rule, $\mathcal{V}_a(\beta_i) \subseteq \mathcal{V}_a(\alpha_i)$ and $\mathcal{V}_f(\beta_i) \subseteq \mathcal{V}_f(\alpha_i)$.

A fixed point rule may be added to the weak $\lambda\sigma$-calculus :

$$\text{(Fix)} \quad (\mu M)[s] \rightarrow M[(\mu M)[s].s]$$

It is then straightforward to add reduction rules for constants to the weak $\lambda\sigma$-calculus, such as integers and booleans, getting exactly the PCF calculus [18, 8]. Provided that, the conditions (1), (2) and (3) above, still hold for the extended system.

A parallel reduction $A\colon M \rightarrow N$ is fully characterized by the set $U$ of the occurrences of the redexes that it contracts. This is written, $A\colon M \xrightarrow{U} N$. If $U$ is empty, then $A$ is the empty reduction $\emptyset$. From now on, parallel reductions are simply called reductions. The derivation relation $\rightarrow^*$ is the reflexive and transitive closure of $\rightarrow$. The length $|A|$ of a derivation $A$ is the number of $\rightarrow$ steps in it. The null derivation $O$ is the derivation of length zero. It is not to be confused with the empty reduction $\emptyset$, since $|\emptyset| = 1$. The set $\mathcal{D}(M)$ is the set of all the derivations starting from term $M$. If $A : M \rightarrow^* N$ and $B : N \rightarrow^* P$, then $AB$ is the derivation $M \rightarrow^* N \rightarrow^* P$.

For instance, consider the Orthogonal labeled system defined by the single rule $I^e(x) \rightarrow x^{<I^e(x),\, b>}$ and by the domain function $dom(I) = \{1\}$. Then, all the derivations starting from the term $I^a(I^a(x))$ are represented below :

$$\begin{tikzcd}
& I^a(I^a(x)) \arrow[dl, "\{\epsilon\}"'] \arrow[dd, "\{\epsilon,1\}"] \arrow[dr, "\{1\}"] & \\
I^{<I^a(x),\, b>.a}(x) \arrow[dr, "\{\epsilon\}"'] & & I^a(x^{<I^a(x),\, b>}) \arrow[dl, "\{\epsilon\}"] \\
& x^{<I^a(x),\, b>.<I^a(x),\, b>} &
\end{tikzcd}$$

**Remark 3.1** *Let $A : M \rightarrow N$ be a reduction in an Orthogonal $T^l RS$. Let $u$ be an occurrence in $N$. We have two cases. Either $u$ is a residual of some occurrence $v$ in $M$, and*

$$\tau(N/u) = \,<\!\alpha_1,\, a_1\!> \ldots <\!\alpha_n,\, a_n\!>.\tau(M/v).$$

$$
\begin{array}{lrl}
\text{(WeakBeta)} & \alpha_1 = (((\lambda M)^{l_3}[s])^{l_2}\ N)^{e_1} & \rightarrow (M^{<\alpha_1,\, c>}[(N^{<\alpha_1,\, d>}.s^{<\alpha_1,\, e>})^{<\alpha_1,\, b>}])^{<\alpha_1,\, a>} = \beta_1 \\
\text{(App)} & \alpha_2 = ((M\ N)^{l_2}[s])^{e_1} & \rightarrow ((M^{<\alpha_2,\, c>}[s^{<\alpha_2,\, d>}])^{<\alpha_2,\, b>}\ (N^{<\alpha_2,\, f>}[s^{<\alpha_2,\, g>}])^{<\alpha_2,\, e>})^{<\alpha_2,\, a>} = \beta_2 \\
\text{(FVar)} & \alpha_3 = (\mathtt{1}^{l_2}[(M.s)^{l_3}])^{e_1} & \rightarrow M^{<\alpha_3,\, a>} = \beta_3 \\
\text{(RVar)} & \alpha_4 = ((\mathtt{n+1})^{l_2}[(M.s)^{l_3}])^{e_1} & \rightarrow (\mathtt{n}^{<\alpha_4,\, b>}[s^{<\alpha_4,\, c>}])^{<\alpha_4,\, a>} = \beta_4
\end{array}
$$

Figure 2: Labeled reduction rules for the weak $\lambda\sigma$-calculus

$$
\text{(id)}\quad M \rightarrow M \qquad\qquad \text{(rec}f\text{)}\quad \frac{M_i \rightarrow N_i \text{ if } i \in dom(f) \qquad M_i = N_i \text{ if } i \notin dom(f)}{f^l(M_1,\, M_2, \ldots M_n) \rightarrow f^l(N_1,\, N_2, \ldots N_n)}
$$

$$
\text{(red)}\quad \frac{\sigma(x) \rightarrow \rho(x) \text{ if } x \in \mathcal{V}_a(\alpha) \qquad \sigma(x) = \rho(x) \text{ if } x \in \mathcal{V}_f(\alpha)}{l.\sigma(\alpha) \rightarrow l.\rho(\beta)}
$$

Figure 3: Definition of parallel reduction

$$
\text{(IdTerms)}\quad M \rightarrow M \qquad\qquad \text{(IdSubst)}\quad s \rightarrow s
$$

$$
\text{(recApp)}\quad \frac{M_1 \rightarrow N_1 \quad M_2 \rightarrow N_2}{(M_1\ M_2)^l \rightarrow (N_1\ N_2)^l} \qquad \text{(recCons)}\quad \frac{M_1 \rightarrow N_1 \quad s_2 \rightarrow t_2}{(M_1.s_2)^l \rightarrow (N_1.t_2)^l} \qquad \text{(recSubst)}\quad \frac{s_2 \rightarrow t_2}{(M_1[s_2])^l \rightarrow (M_1[t_2])^l}
$$

$$
\text{(redBeta)}\quad \frac{\sigma(N) \rightarrow \rho(N) \quad \sigma(s) \rightarrow \rho(s) \quad \sigma(M) = \rho(M)}{l.\sigma(\alpha_1) \rightarrow l.\rho(\beta_1)} \qquad \text{(redApp)}\quad \frac{\sigma(s) \rightarrow \rho(s) \quad \sigma(M) = \rho(M) \quad \sigma(N) = \rho(N)}{l.\sigma(\alpha_2) \rightarrow l.\rho(\beta_2)}
$$

$$
\text{(redFVar)}\quad \frac{\sigma(M) \rightarrow \rho(M) \quad \sigma(s) \rightarrow \rho(s)}{l.\sigma(\alpha_3) \rightarrow l.\rho(\beta_3)} \qquad \text{(redRVar)}\quad \frac{\sigma(M) \rightarrow \rho(M) \quad \sigma(s) \rightarrow \rho(s)}{l.\sigma(\alpha_4) \rightarrow l.\rho(\beta_4)}
$$

Figure 4: Parallel reductions for the weak $\lambda\sigma$-calculus

*Or $u$ is created by the contraction of some redex $l.\sigma(\alpha)$ by $A$, and*

$$\tau(N/u) = <\alpha_1, a_1> \ldots <\alpha_n, a_n>.<\alpha, a>.$$

*In both cases, the $\alpha_i$'s are the names of redexes contracted by $A$, and the $a_i$'s are atomic labels from the corresponding $\beta_i$'s. And, in the second case, $a$ is some atomic label in the right-hand side $\beta$ of the reduction rule $\alpha \rightarrow \beta$.*

## 3.1 The parallel moves lemma

The parallel moves lemma is the fundamental property of Orthogonal TRS, as studied in [9]. It also holds for Orthogonal T$^l$RS, the key point being that redexes do not overlap in terms. We give a full proof of the strong confluence of parallel reductions, as this proof illustrates some important points in our labeling scheme. First, two technical lemmas.

**Lemma 3.2** *Let $M$ be a labeled term. Let $\sigma$ and $\rho$ be two substitutions over labeled terms. We have :*

$$\left\{ \begin{array}{l} \forall x_a \in \mathcal{V}_a(M),\ \sigma(x_a) \rightarrow \rho(x_a) \\ \forall x_f \in \mathcal{V}_f(M),\ \sigma(x_f) = \rho(x_f) \end{array} \right. \Rightarrow \sigma(M) \rightarrow \rho(M)$$

**Proof**: Easy induction over $|M|$. □

**Lemma 3.3** *Let $A : M \rightarrow N$. For any label $l$, there exists a reduction $B : l.M \rightarrow l.N$.*

**Proof**: The only subtle case is (red). In this case there exist a label $m$, two substitutions $\sigma$ and $\rho$, satisfying the premises of rule (red) and a reduction rule $\alpha \rightarrow \beta$, such that $M = m.\sigma(\alpha)$ and $N = m.\rho(\beta)$. Let $\gamma$ be either $\alpha$ or $\beta$. By the associativity of ".", $l.(m.\sigma(\gamma))$ may be written $(l.m).\sigma(\gamma)$. And $l.M \rightarrow l.N$ is proved by :

$$\text{(red)}\ \frac{A_a : \sigma(x_a) \rightarrow \rho(x_a) \qquad \sigma(x_f) = \rho(x_f)}{B : (l.m).\sigma(\alpha) \rightarrow (l.m).\rho(\beta)} \quad \square$$

**Proposition 3.4 (Strong confluence)** *Let $M$ be a labeled term. Let $A : M \rightarrow N$ and $B : M \rightarrow P$. There exists two reductions $C$ and $D$ such that $C : N \rightarrow Q$ and $D : P \rightarrow Q$.*

**Proof**: By induction on the proof trees of reductions $A$ and $B$. The non-trivial cases are :

- (rec $\leftrightarrow$ rec) Direct use of the induction hypothesis.
- (red $\leftrightarrow$ rec) Let $A$ and B be of the form :

$$\text{(red)}\ \frac{A_a : \sigma(x_a) \rightarrow \rho(x_a) \qquad \sigma(x_f) = \rho(x_f)}{A : l.\sigma(\alpha) \rightarrow l.\rho(\beta)}$$

$$\text{(rec)}\ \frac{\ldots}{B : M \rightarrow P}$$

  By the non-overlapping condition, we have $P = l.\varphi(\alpha)$ with $\forall x_a \in \mathcal{V}_a(\alpha)\ B_a : \sigma(x_a) \rightarrow \varphi(x_a)$ and $\forall x_f \in \mathcal{V}_f(\alpha)$, $\sigma(x_f) = \varphi(x_f)$. By induction hypothesis, for any variable $x_a$ in $\mathcal{V}_a(\alpha)$, there exists a term $Q_a$ such that $\rho(x_a) \rightarrow Q_a \leftarrow \varphi(x_a)$. Define the substitution $\psi$ by :

$$\begin{array}{ll} \psi(x_a) = Q_a & \text{if } x_a \in \mathcal{V}_a(\alpha) \\ \psi(x_f) = \sigma(x_f) & \text{if } x_f \in \mathcal{V}_f(\alpha) \\ \psi(x) = x & \text{otherwise} \end{array}$$

  Let $Q$ be the term $l.\psi(\beta)$. The inference rule (red) may be applied directly to prove $P = l.\varphi(\alpha) \rightarrow Q = l.\psi(\beta)$. Moreover, since $\mathcal{V}_a(\beta) \subseteq \mathcal{V}_a(\alpha)$ and $\mathcal{V}_f(\beta) \subseteq \mathcal{V}_f(\alpha)$, lemma 3.2 does apply and $\rho(\beta) \rightarrow \psi(\beta)$. Finally, $N = l.\rho(\beta) \rightarrow Q = l.\psi(\beta)$ comes from lemma 3.3.

- (red $\leftrightarrow$ red) By the non-overlapping condition, the reduction rules involved in $A$ and $B$ are, in fact, the same rule $\alpha \rightarrow \beta$ :

$$\text{(red)}\ \frac{A_a : \sigma(x_a) \rightarrow \rho(x_a) \qquad \sigma(x_f) = \rho(x_f)}{A : l.\sigma(\alpha) \rightarrow l.\rho(\beta)}$$

$$\text{(red)}\ \frac{B_a : \sigma(x_a) \rightarrow \varphi(x_a) \qquad \sigma(x_f) = \varphi(x_f)}{B : l.\sigma(\alpha) \rightarrow l.\varphi(\beta)}$$

  The proof is then similar to the one above. □

Let $A : M \xrightarrow{U} N$ and $B : M \xrightarrow{V} P$. Then, thanks to the non-overlapping condition and because the residuals of an allowed occurrence are allowed occurrences, all the subterms rooted at occurrences in $V\backslash A$ are redexes. They are residual redexes. And the reduction $B\backslash A : N \xrightarrow{V\backslash A} Q$ is the residual reduction of $B$ by $A$. We use $A \sqcup B$ to denote the derivation $A(B\backslash A)$.

**Lemma 3.5 (The parallel moves lemma)** *Let $M$ be a labeled term. Let $A$ and $B$ be two reductions starting from $M$ :*

*(1) Both $A \sqcup B$ and $B \sqcup A$ lead to the same term.*

*(2) For any occurrence $u$ in $M$, $u\backslash(A \sqcup B) = u\backslash(B \sqcup A)$.*

**Proof**: By a careful relabeling of $M$ one may "track" the redexes contracted by $A$ and $B$, as any other set of occurrences in $M$. The result then follows from the strong confluence of parallel reductions. □

The derivations $A \sqcup B$ and $B \sqcup A$, satisfying (1) and (2), are almost equivalent derivations. A key consequence of the parallel moves lemma is the extension of the notion of residual to composite derivations. As shown by figure 5, the parallel moves lemma for derivations $A$ and $B$ of any lengths $n$ and $m$ comes from the parallel moves lemma for the elementary reduction steps. More precisely, the residual derivations are defined inductively :

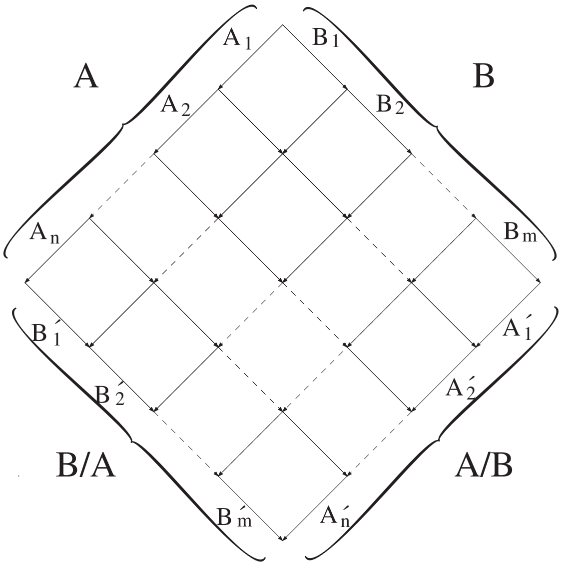



Figure 5: The parallel moves lemma for derivations

(1) $0\backslash B = 0$

(2) $(A_1A_2)\backslash B = (A_1\backslash B)(A_2\backslash(B\backslash A_1))$

(3) $A\backslash 0 = A$

(4) $A\backslash(B_1B_2) = (A\backslash B_1)\backslash B_2$

This extended notion of residual yields an equivalence relation beetween derivations : Let $A$ and $B$ be two derivations starting from the same term $M$. The derivations $A$ and $B$ are equivalent, or $A \equiv B$, iff, for all derivations $C$, starting from $M$, $C\backslash A = C\backslash A$. It can be shown that, for any two derivations $A$ and $B$, starting from the same term, we have $A \sqcup B \equiv B \sqcup A$. This result is known as the "cube lemma". The equivalence $\equiv$ behaves well, with respect to the concatenation and residual operations. More precisely, let $A$ and $B$ be two equivalent derivations, starting from the term $M$, leading to the term $N$, then we have :

$$\begin{array}{lll} \forall C & \text{starting from } M, & A\backslash C \equiv B\backslash C \\ \forall C & \text{starting from } N, & AC \equiv BC \\ \forall C & \text{leading to } M, & CA \equiv CB \end{array}$$

There are simple decision procedures for the inclusion $\sqsubseteq$ and the equivalence $\equiv$. For any two derivations $A$ and $B$ starting from the same term :

$$A \sqsubseteq B \Leftrightarrow A\backslash B = \emptyset^{|A|} \quad A \equiv B \Leftrightarrow B \sqsubseteq A \sqsubseteq B$$

We shall not go further into the development of the properties of derivations. The study in [9] is applicable to Orthogonal $\mathrm{T}^l$RS, since all results follow from the parallel moves lemma. We mention two useful results.

(1) The relation $\rightarrow^*$ is Church-Rosser, therefore normal forms, when they exist, are unique.

(2) $<\mathcal{D}(M)/, \equiv, \sqsubseteq, \sqcup>$ is an upper semi-lattice.

Proposition (1) is a classical result. Proposition (2) follows from algebraic manipulations and inductions over definitions.

# 4 Complete derivations

The complete derivations are well behaved parallel derivations. They mimic the derivations on graphs, by contracting at once all the instances of a shared redex.

## 4.1 Redex family

Recall that a redex is in the form $l.\sigma(\alpha)$ where $l$ is a label, $\sigma$ is a substitution and $\alpha$ is the name of the redex.

**Definition 4.1 (Redex family)** *Let $M$ be a labeled term. A* redex family *$\mathcal{F}$ in $M$ is any set of redexes with the same name. A* complete redex family *$\overline{\mathcal{F}}$ in $M$ is the set of all redexes with the same name.*

The intuition is that the name of a redex fully characterizes it. Let $A\colon M \rightarrow^* N$ be a derivation. If a redex $S$ in $N$ is a residual of a redex $R$ in $M$, then $R$ and $S$ have the same name. Furthermore, the labels inside the name of a created redex tells us about its history :

**Definition 4.2 (The contribution relation)** *Let $\alpha$ and $\gamma$ be two redex names. The redex name $\alpha$ directly contributes to $\gamma$, written $\alpha < \gamma$, iff there is a label $l$ in $\gamma$ such that :*

$$l = l_1.{<}\alpha,\, a{>}.l_2$$

*where $l_1$, $l_2$ are possibly empty labels, and $a$ is an atomic label. The* contribution *relation $<^+$ is the transitive closure of $<$. The "is equal or contributing" relation $<^*$ is the reflexive and transitive closure of $<$.*

In fact, in the definition of direct contribution, $\gamma$ could be any term $M$. We thereby define the direct contribution relation $\alpha < M$ for any redex name $\alpha$ and any term $M$. The relation $<^+$ and $<^*$ are also extended to the case where $\gamma$ is any term $M$. For any redex name $\alpha$ and any term $M$ :

| | |
|---|---|
| $\alpha <^+ M \Leftrightarrow$ | $\exists l = l_1.{<}\gamma,\, a{>}.l_2$ label in $M$ with $\alpha <^* \gamma$ |
| $\alpha <^* M \Leftrightarrow$ | $\alpha <^+ M$ or $\alpha$ is the name of a redex in $M$ |

Notice that the extended $<^+$ and $<^*$ relations are no more transitive and that the extended $<^*$ relation is no more a reflexive relation.

The following lemma explicits how the labels capture the "creators" of a new redex.

**Lemma 4.3** *Let $A: M \to N$ be a reduction. Let $\alpha$ be the name of a redex $R$ in $N$. Either, there exists a redex $S$ in $M$, contracted by $A$, of name $\gamma$ directly contributing to $\alpha$, or $R$ is a residual of a redex of name $\alpha$ in $M$.*

**Proof**: Consider a redex $R = l.\sigma(\alpha)$, of name $\alpha$, at an allowed occurrence $u$ in $N$. Suppose that there is no redex name $\gamma$ in $M$ such that $\gamma < \alpha$. Notice that $\tau(R) = l.e$ where $e = \tau(\alpha)$. The occurrence $u$ cannot be a created occurrence. If it was the case, then $e$ would be equal to $<\gamma, a>$ where $\gamma$ is a redex name in $M$. Therefore the occurrence $u$ in $N$ is a residual of an allowed occurrence $v$ in $M$. Now, what remains to be proved is that $M/v$ is a redex of name $\alpha$, of which $R$ is a residual. That is to say, for any non variable occurrence $w \succ \epsilon$ in $\alpha$, the occurrence $vw$ is a valid occurrence in $M$, the occurrence $uw$ in $N$ is a residual occurrence of $vw$ and $\tau(M/vw) = \tau(N/uw)$. For the same reason as the occurrence $u$, the occurrence $uw$ cannot be a created occurrence. Therefore $uw$ is a residual of some occurrence $vw'$, below $v$, in $M$. Let state $m = \tau(N/uw)$ and $m' = \tau(M/uw')$. *A priori*, we have $m = <\alpha_1, a1> \ldots <\alpha_n, a_n> .m'$, where the $\alpha_i$'s are the names of redexes in $M$. But, since no redex name in $M$ can directly contribute to $\alpha$, the labels $m$ and $m'$ must be equal. Therefore, there are no redexes to be contracted in $M$, between occurrences $v$ and $vw'$, and $w = w'$. So, there is a redex of name $\alpha$ at occurrence $v$ in $M$, of which $R$ is a residual. □

The direct extension to derivations of this lemma will prove useful.

**Lemma 4.4** *Let $A: M \to^* N$ be a derivation. Let $\alpha$ be the name of a redex $R$ in $N$. Either, there exists a redex $S$ of name $\gamma$ in $M$, such that $\gamma$ contributes to $\alpha$, and a residual of $S$ is contracted by some step in $A$. Or, $R$ is a residual by $A$ of a redex of name $\alpha$ in $M$.*

## 4.2 Complete derivations

**Definition 4.5 (Complete reduction)** *Let $A$ be a reduction, contracting the set of redex occurrences $U$. Let $\{\mathcal{F}_1, \mathcal{F}_2, \ldots \mathcal{F}_n\}$ be the partition by families of the set $U$. The completion $\overline{A}$ of $A$ is the reduction contracting the set of redex occurrences $\overline{U} = \overline{\mathcal{F}_1} \cup \overline{\mathcal{F}_2} \cup \ldots \overline{\mathcal{F}_n}$.*

Note that $A \sqsubseteq \overline{A}$, since $U \subseteq \overline{U}$. The completion of derivations is defined inductively :

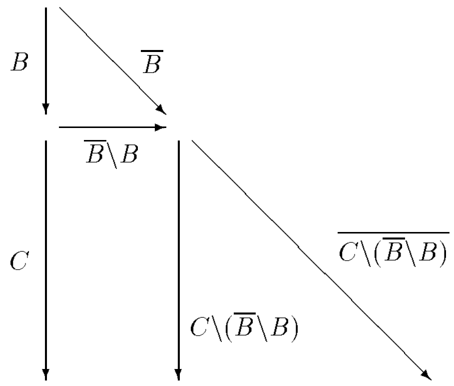



Figure 6: Completion of derivation BC

$$\overline{A} = \overline{B}\,(\overline{C\backslash(\overline{B}\backslash B)})$$

A derivation is complete, iff it contracts only complete families at each step ($\overline{A} = A$). Complete derivations are related to ordinary derivations.

**Proposition 4.6** *Let $A$ be a derivation. Then, $A \sqsubseteq \overline{A}$.*

**Proof**: To get an idea, see figure 6. The actual proof uses the definition $A \sqsubseteq B \Leftrightarrow \exists C,\ AC \equiv B$. □

Complete derivations are closed under the concatenation operation.

**Lemma 4.7** *Let $\overline{A}: M \to^* N$ and $\overline{B}: N \to^* P$ be two complete derivations. The derivation $\overline{A}\,\overline{B}$ is complete.*

Conversely, if a derivation $AB$ is complete then $A$ and $B$ are also complete.

**Definition 4.8 (Consistency)** *A term $M$ is consistent, iff, for any redex in $M$, of name $\alpha$, the name $\alpha$ does not contribute to $M$.*

This technical condition expresses the fact that a redex family should not coexist in the same term, with subterms which it could have helped to generate. A very useful consequence of consistency is that residual families are characterized by their names :

**Lemma 4.9** *Let $M$ be a consistent term. Let $A: M \to^* N$. Let $\overline{\mathcal{F}}$ be a complete redex family in $M$, of name $\alpha$. The set of residual occurrences $\overline{\mathcal{F}}\backslash A$ is exactly the complete redex family of name $\alpha$ in $N$ :*

*$\overline{\mathcal{F}}\backslash A = \{u$ with $N/u$ redex of name $\alpha$ in $N\}$.*

**Proof**: Only the inclusion $\supseteq$ is not a direct consequence of the definitions. Let $R$ be a redex, of name $\alpha$, at occurrence $u$ in $N$. Suppose $R$ is not a residual of a redex in $M$. Then, by lemma 4.4, there exist a redex name $\gamma$ in $M$, contributing to $\alpha$. This contradicts the consistency of $M$. $\square$

**Remark 4.1** *The complete derivations starting from a consistent term $M$ well behave : the names of the families contracted are all different one from another. More precisely, let $\overline{\mathcal{F}}$ be a complete redex family of name $\alpha$ in $M$. Let $\overline{A} = \overline{A_1}\,\overline{A_2}\,\overline{A_3}$ be a complete derivation starting from $M$, such that $\overline{A_2}$ is a complete reduction contracting a family of name $\alpha$. Then, the family of name $\alpha$, contracted by $\overline{A_2}$ is $\overline{\mathcal{F}}\backslash\overline{A_1}$. And no step in the derivation $\overline{A_3}$ contracts a family of name $\alpha$. In such case, by abuse of language, we say that $\overline{A}$ contracts $\overline{\mathcal{F}}$.*

Complete derivations are compatible with the consistency condition.

**Lemma 4.10 (Preservation of consistency)** *Let $M$ be a consistent term. Let $\overline{A} : M \rightarrow^* N$ be a complete derivation. The term $N$ is consistent.*

**Proof**: It is sufficient to prove the lemma for $|\overline{A}| = 1$. Let $\alpha$ be the name of a redex $R$ in $N$. Suppose that $\alpha$ contributes to $N$, then there exists an occurrence $u$ in $N$ and a label $l = \tau(N/u) = l_1.{<}\gamma, a{>}.l_2$, with $\alpha <^* \gamma$. The redex $R$ may be a residual of a redex in $M$, or not. The occurrence $u$ may be a residual of an occurrence or not. Therefore, there are four cases :

(1) The redex $R$ is a residual. Then $\alpha$ is the name of a complete family in $M$, and this family is not contracted by $\overline{A}$ complete.

  (i) The occurrence $u$ is the residual by $\overline{A}$ of some occurrence $v$ in $M$, and $l = {<}\alpha_1, a_1{>} \ldots {<}\alpha_n, a_n{>}.\tau(M/v)$. The names $\alpha_i$ for $i \in [1\ldots n]$ are the names of redexes contracted by $\overline{A}$, therefore $\alpha$ cannot be equal to any of the $\alpha_i$'s. Furthermore $\alpha <^+ \alpha_i$ for some $i \in [1\ldots n]$ violates the consistency of $M$, since the labels in the $\alpha_i$'s are labels in $M$. Therefore $\tau(M/v)$ must in the form $m_1.{<}\gamma, a{>}.m_2$, with $\alpha <^* \gamma$, which, again, contradicts the consistency of $M$.

  (ii) The occurrence $u$ is created by the contraction of a redex of name $\delta$. Then, by a proof similar to (i) above, $\alpha <^+ N \Rightarrow \alpha <^* \delta \Rightarrow \alpha <^* M$.

(2) The redex $R$ is not a residual. Then, by lemma 4.3, there exists a family in $M$, of name $\eta$, contracted by $\overline{A}$, such that $\eta < \alpha$.

  (i) We have $u \in v\backslash\overline{A}$ and $l = {<}\alpha_1, a_1{>} \ldots {<}\alpha_n, a_n{>}.\tau(M/v)$. If $\alpha <^* \alpha_i$ for some $i \in [1\ldots n]$, then we have $\eta <^+ \alpha_i$, which violates the consistency of $M$. Moreover, $\alpha <^* \tau(M/v)$ implies $\delta <^+ \tau(N/v)$ and contradicts the consistency of $M$.

  (ii) The occurrence $u$ is created by the contraction of a redex of name $\delta$. Then, we have $\alpha <^+ M \Rightarrow \alpha <^* \delta \Rightarrow \eta <^+ \delta \Rightarrow \eta <^+ M$. $\square$

The result of lemma 4.9 on families is also a result on reductions. It extends to composite derivations :

**Lemma 4.11** *Let $M$ be a consistent term. Let $\overline{A} : M \rightarrow^* N$ be a complete derivation and $B : M \rightarrow^* P$ a derivation. The derivation $\overline{A}\backslash B$ is complete.*

**Proof**: The proof is by induction on $|\overline{A}|$, thanks to the preservation of consistency lemma (4.10). $\square$

Let $\overline{\mathcal{D}}(M)$ be the set of the complete derivations starting from a consistent term $M$. Lemmas 4.7 and 4.11 implies that ${<}\overline{\mathcal{D}}(M)/, \equiv, \sqsubseteq, \sqcup{>}$ is a sublattice of $\mathcal{D}(M)$, with the same lub function.

## 4.3 Cost of a derivation

**Definition 4.12 (Cost of a reduction)** *Let $A$ be a reduction contracting the set of redex occurrences $U$. The* cost *$\mathcal{C}(A)$ of $A$ is the number of different redex families in $U$.*

For a composite derivation, $\mathcal{C}(AB) = \mathcal{C}(A) + \mathcal{C}(B)$. Intuitively, we assume that the cost of the reduction of a given family is one. From an implementation point of view, this means that we have to represent a complete family by a shared data structure. The cost measure is relative to this data structure, and it will be of practical interest only if this data structure is simple enough.

The following results on costs are intuitive and relatively easy, but they deserve some attention, since they illustrate the relevance of our cost measure.

**Lemma 4.13 (Cost of residual)** *Let $A : M \rightarrow^* N$ and $B : M \rightarrow^* P$ be two derivations, starting from the same term. We have : $\mathcal{C}(A\backslash B) \leq \mathcal{C}(A)$.*

**Proof**: By induction on $|A|$.

- If $|A| = 1$, then consider the (may be not complete) families contracted by $A$ : $\mathcal{F}_1, \mathcal{F}_2 \ldots \mathcal{F}_a$. The reduction $A\backslash B$ contracts the non-empty families among $\mathcal{F}_1\backslash B, \mathcal{F}_2\backslash B, \ldots \mathcal{F}_a\backslash B$. The result follows.

- Now, if $A$ is a composite derivation $A_1A_2$, then $\mathcal{C}(A\backslash B) = \mathcal{C}(A_1\backslash B) + \mathcal{C}(A_2\backslash(B\backslash A_1))$. The result follows by direct application of the induction hypothesis. $\square$

**Proposition 4.14 (Cost of completion)** *Let $A$ be any derivation. We have : $\mathcal{C}(\overline{A}) \leq \mathcal{C}(A)$.*

**Proof**: By induction on the definition of completion. Let $A = BC$. Then by definition, $\mathcal{C}(\overline{A}) = \mathcal{C}(\overline{B}) + \mathcal{C}(\overline{C\backslash(\overline{B}\backslash B)})$. The induction hypothesis gives $\mathcal{C}(\overline{A}) \leq \mathcal{C}(B)+\mathcal{C}(C\backslash(\overline{B}\backslash B))$. Therefore, by Lemma 4.13 $\mathcal{C}(\overline{A}) \leq \mathcal{C}(B) + \mathcal{C}(C)$.□

Together with $A \sqsubseteq \overline{A}$, this proves that contracting a complete family whenever one of its members is contracted goes (possibly) further at a (possibly) lower cost. This result is an important step toward the search of an optimal derivation, as it enables to restrict ourselves to the sublattice of complete derivations.

# 5 The implementation of complete derivations

In this section, it is shown that complete reductions may be implemented as reductions on directed acyclic graphs.

The idea of representing DAGs as terms labeled by their location in a graph structure was introduced in section 2. But the labeling scheme of section 3 does not fit this purpose. The labels retain too much of the history of a subterm. For instance, the residuals of a redex may very well have different top labels, although they should be shared in a graph implementation.

Therefore, graphs are represented by a restricted labeling scheme. This new scheme is build with the same set $\mathcal{A}$ of *atomic labels* as $\mathcal{T}^l$. We defined the set of pre-DAGs, $\mathcal{T}^d$ as :

$$\begin{array}{lll} \mathcal{T}^d: & \text{pre-DAGs} & M ::= x \mid f^l(M_1, M_2, \ldots M_n) \\ \mathcal{L}^d: & \text{locations} & l ::= a \mid l.a \end{array}$$

There is an homomorphism $H$, from the labeled terms in $\mathcal{T}^l$ to the pre-DAGs in $\mathcal{T}^d$ :

$$\left\{ \begin{array}{rcl} H(x^l) &=& x \\ H(f^l(\ldots, M_i, \ldots)) &=& f^{h(l)}(\ldots, H(M_i), \ldots) \end{array} \right.$$

$$\left\{ \begin{array}{rcl} h(a) &=& a \\ h(l.e) &=& h(e) \\ h(<\alpha, a>) &=& h(\tau(\alpha)).a \end{array} \right.$$

Let $M$ be a labeled term in $\mathcal{T}^l$. The term $H(M)$ in $\mathcal{T}^d$ is the *pre-DAG representation* of $M$. The simple label $\&(M) = h(\tau(M))$ is the *location* of $M$. In the definition of $h$, the last clause may be rewritten as $h(<\alpha, a>) = \&(\alpha).a$. Pre-DAGs are DAGs, if and only if they meet the (intuitive) definition 2.1.

The DAG condition on labeled terms is different. It is weaker than the intuitive DAG definition and stronger than the "$H(M)$ is a DAG in $\mathcal{T}^d$" condition.

**Definition 5.1 (DAG in labeled terms)** *Let $M$ be a labeled term. The term $M$ is a DAG, iff, for any two subterms of $M$, that is $M_1 = f_1^{l_1.e_1}(M_1^1, M_2^1, \ldots M_{n_1}^1)$ and $M_2 = f_2^{l_2.e_2}(M_1^2, M_2^2, \ldots M_{n_2}^2)$, we have :*

$$\begin{array}{c} \&(M_1) = \&(M_2) \\ \Downarrow \\ M_1 \sim M_2 \;\Leftrightarrow\; \left\{ \begin{array}{l} f_1 = f_2 \\ e_1 = e_2 \\ \forall i \in [1 \ldots n_{n_1}]\ M_i^1 = M_i^2 \end{array} \right. \end{array}$$

This definition makes sense in terms of implementation. Indeed, if $M$ is a DAG in $\mathcal{T}^l$, then its representation $H(M)$ is a DAG in $\mathcal{T}^d$. Furthermore, let $R_1$ and $R_2$ be two redexes in $M$, of names $\alpha_1$ and $\alpha_2$. Then, $R_1$ and $R_2$ are in the same family —$\alpha_1 = \alpha_2$—, iff they have the same location —$\&(R_1) = \&(R_2)$—.

This last point is important. Intuitively, the reductions on DAGs are the reductions, that are "complete" relatively to the locations of the redexes.

**Definition 5.2 (DAG-reductions)** *Let $M \in \mathcal{T}^l$. Let $A : M \xrightarrow{U} M$. The reduction $A$ is a DAG-reduction, iff, for any two redex occurrences, $u_1$ and $u_2$, in $M$ : $u_1 \in U$ and $\&(M/u_1) = \&(M/u_2)$ implies $u_2 \in U$.*

From the previous remark, the DAG-reductions starting from a DAG are *exactly* the complete reductions.

Also notice that, if all the labels in $M \in \mathcal{T}^l$ are atomic, then $M$ and $H(M)$ are syntactically equal, and then, the complex DAG definition is equivalent to the intuitive definition.

A general T$^l$RS $\Sigma$ is said to be a DAG Rewriting System (DRS), iff the right-hand-sides of the reduction rules in $\Sigma$ (the $\beta_i$'s) are DAGs. This restriction makes sense, since non-DAG right-hand sides may be expected to introduce non-DAG parts in the result of a reduction, even if reduction is started from a DAG. Moreover, this condition is easily checked, by only looking at the atomic labels in the $\beta_i$'s. All the systems seen so far are DRS.

**Lemma 5.3 (Preservation of DAGs)** *In a DRS, let $M$ be a consistent DAG in $\mathcal{T}^l$. Let $\overline{A} : M \rightarrow^* N$ be complete derivation. Then $N$ is a DAG.*

**Proof**: Let $N/u_1$ and $N/u_2$ be two subterms of $N$, with $\&(N/u_1) = \&(N/u_2)$. There are three cases :

(1) Both occurrences $u_1$ and $u_2$ are residuals from occurrences in $M$,i.e. $u_1 \in \{v_1\}\backslash\overline{A}$ and $u_2 \in \{v_2\}\backslash\overline{A}$. Then, by the nature of reduction, $\&(M/v_1) = \&(N/u_1)$ and $\&(M/v_2) = \&(N/u_2)$. Therefore, because $M$ is a DAG : $M/v_1 \sim M/v_2$. The derivation $\overline{A}$ being complete, the the redexes contracted in $M/v_1$ and $M/v_2$ are the same. Hence $N/u_1 \sim N/u_2$.

(2) The occurrence $u_1$ created by the contraction of a redex of name $\alpha$, whereas $u_2 \in \{v_2\} \backslash \overline{A}$. Then, there exists a redex of name $\alpha$ in $M$ directly contributing to $\tau(M/v_2)$. So, this case is excluded by the consistency of $M$.

(3) Both $u_1$ and $u_2$ are created by $\overline{A}$. Let $R_1 = l_1.\sigma_1(\alpha_1)$ and $R_2 = l_2.\sigma_2(\alpha_2)$ be the two redexes in $M$ of which contractions create $u_1$ and $u_2$. Let $v_1$ and $v_2$ be their occurrences. The created occurrence $u_1$ is equal to $v_1' w_1$, where $v_1'$ is a "pseudo-residual" of $v_1$ and $w_1$ is a non-variable occurrence in $\beta_1$, with $\tau(\beta_1/w_1) = <\alpha_1, a_1>$. Similarly $u_1 = v_2' w_2$. Then, the top labels of $N/u_1$ and $M/u_2$ are in the form : $\tau(N/u_1) = m_1.<\alpha_1, a_1>$ and $\tau(N/u_2) = m_2.<\alpha_2, a_2>$. From $\&(N/u_1) = \&(N/u_2)$, we get $\&(\alpha_1).a_1 = \&(\alpha_2).a_2$, From $\&(\alpha_1) = \&(\alpha_2)$ and because $M$ is a DAG, we get $\alpha_1 = \alpha_2 = \alpha$ (and $\beta_1 = \beta_2 = \beta$). Therefore, $\&(R_1) = \&(R_2) = \&(\alpha)$. And, in fact, because $M$ is a DAG : $R_1 = l_1.\sigma(\alpha)$ and $R_2 = l_2.\sigma(\alpha)$ (1). We also have $a_1 = a_2 = a$, and therefore $\&(\beta/w_1) = \&(\beta/w_2)$. But $\beta$ is a DAG, and we get $\beta/w_1 = \beta/w_2(2)$. By (1), (2), and because $\overline{A}$ is complete we get $N/u_1 = \rho(\beta/w_1) = N/u_2 = \rho(\beta/w_2)$ where $\rho$ is the substitution carrying the reductions performed below $R_1$ and $R_2$.

$\square$

This result enables to identify the DAG-derivations and the complete derivations and closes the study of the implementation of complete derivations.

# 6 Complete-call-by-need derivations

In this section, we introduce a new notion of call-by-need derivation among the complete derivations. It is based on a straightforward criterion on labels. Unfortunately, there does not seem to be a simple counterpart of this notion of call-by-need in unlabeled systems.

## 6.1 Properties of complete-call-by-need derivations

As the redex families are identified by their names, a family contributes to a complete derivation $\overline{A}$ when its name is present in the labels of the final term of $\overline{A}$ :

**Definition 6.1 (Contributing family)** *Let $M$ be a term. Let $\overline{A} : M \rightarrow^* N$. Let $\overline{\mathcal{F}}$ be a complete redex family of name $\alpha$ in $M$. The family $\overline{\mathcal{F}}$ is said to contribute to $\overline{A}$, iff its name $\alpha$ contributes to $N$.*

The following syntactical lemmas deal with the origin of contributing names.

**Lemma 6.2** *Let $A : M \rightarrow N$ be a reduction. Let $\alpha$ be a redex name contributing to $N$. Then, either $\alpha$ contributes to $M$, or $\alpha$ is the name of a redex in $M$, which is contracted by $A$.*

**Proof**: If $\alpha$ contribute to $N$, then there exists a label $l$ in $N$ with $l = l_1.<\gamma, a> l_2$, where $l_1$ and $l_2$ are (possibly empty) labels, $a$ is an atomic label, and $\gamma$ is a redex name such that $\alpha <^* \gamma$. The label $l$ has been generated by the reduction $A$. From remark 3.1, one can deduce that either $\gamma$ directly contribute to $M$, or $\gamma$ is the name of a redex contracted by $A$. In the first case $\alpha$ contributes to $M$. In the latter case, either $\gamma$ is equal to $\alpha$, or $\alpha$ contributes to $\gamma$, and, therefore, $\alpha$ contributes to $M$. $\square$.

The transitive closure of this lemma easily follows from lemma 4.4 :

**Lemma 6.3** *Let $A : M \rightarrow^* N$ be a derivation. Let $\alpha$ be a redex name contributing to $N$. Then, either $\alpha$ contributes to $M$, or there exists a redex $R$, of name $\gamma$ in $M$, such that $\gamma$ contributes to $\alpha$ or $\gamma$ is equal to $\alpha$, and a residual of $R$ is contracted by some step in $A$.*

Some contributing families are more important than others, in the sense that their names cannot be deleted :

**Definition 6.4 (Needed family)** *Let $M$ be a term and $\overline{\mathcal{F}}$ be a complete family of name $\alpha$ in $M$. Let $\overline{A} : M \rightarrow^* N$ be a complete derivation. The family $\overline{\mathcal{F}}$ is needed by the derivation $\overline{A}$, or $\overline{\mathcal{F}} \in \mathcal{N}(\overline{A})$, iff, for all complete derivations $\overline{B} : N \rightarrow^* P$, the name $\alpha$ contributes to $P$.*

In other words, a family is needed by a complete derivation $\overline{A}$, when it contributes to all the complete derivations greater that $\overline{A}$. Two basic properties of needed families are consequences of the definition :

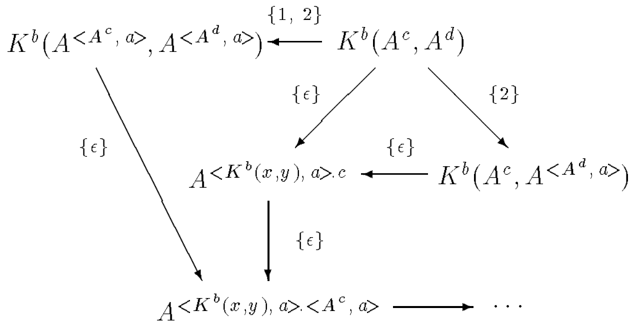
$$\overline{A} \sqsubseteq \overline{B} \Rightarrow \mathcal{N}(\overline{A}) \subseteq \mathcal{N}(\overline{B})$$
$$\overline{A} \equiv \overline{B} \Rightarrow \mathcal{N}(\overline{A}) = \mathcal{N}(\overline{B})$$

A complete-call-by-need derivation (CCBN) is a complete derivation contracting only families needed by the rest of itself, at each step.

**Definition 6.5 (CCBN)** *A complete derivation $\overline{A}$ : $M \rightarrow^* N$ is complete-call-by-need (CCBN), iff, either $\overline{A}$ is the null derivation, or $\overline{A} = \overline{B}\,\overline{C}$, where $\overline{C}$ is complete-call-by-need, and $\overline{B}$ is a complete reduction such that :*

$$\overline{B} : M \overset{\overline{\mathcal{F}_1} \cup \overline{\mathcal{F}_2} \cup \ldots \overline{\mathcal{F}_n}}{\longrightarrow} P \text{ and } \overline{\mathcal{F}_1}, \overline{\mathcal{F}_2}, \ldots \overline{\mathcal{F}_n} \text{ in } \mathcal{N}(\overline{A}).$$

For instance, consider the terms formed with the nullary symbol $A$, the binary symbol $K$, and the variables $x$ and $y$. Also consider the rules $\{A^e \to A^{<A^e, a>}, K^e(x,y) \to x^{<K^e(x,y), b>}\}$ and the domain function $dom(K) = \{1, 2\}$. Then, any derivation contracting the $A^d$ redex (always at occurrence 2), is not CCBN, whereas any derivation not contracting it is.



It is easy to check, from $\overline{A} \sqsubseteq \overline{B} \Rightarrow \mathcal{N}(\overline{A}) \subseteq \mathcal{N}(\overline{B})$, that CCBN derivations derivations are closed by concatenation . The converse $\overline{A_1}\,\overline{A_2}$ CCBN $\Rightarrow \overline{A_1}$ CCBN is not so obvious. It requires an important technical lemma, which we express in a very general form :

**Lemma 6.6** *Let $M$ be a consistent term, $\overline{A}$: $M \to^* N$ and $\overline{B}$: $M \to^* P$ be two complete derivations. Let $\overline{\mathcal{F}}$ be a complete redex family in $M$, of name $\alpha$, needed by $\overline{A}$. Then, either $\overline{\mathcal{F}}$ is contracted by $\overline{B}$ and $\overline{\mathcal{F}}$ is needed by $\overline{B}$, or $\overline{\mathcal{F}}$ is not contracted by $\overline{B}$ and $\overline{\mathcal{F}}\backslash\overline{B}$ is not empty.*

**Proof**: Suppose that there exists a complete derivation $\overline{C}$ : $P \to^* Q$, such that $\alpha$ does not contribute to $Q$. The Church-Rosser property is true for complete derivations (end of section 4.2). Therefore, there exist two complete derivations closing the "diamond diagram" : $\overline{D}$: $Q \to^* R$ and $\overline{E}$: $N \to^* R$ (figure 7). Since $\overline{\mathcal{F}} \in \mathcal{N}(\overline{A})$, the name $\alpha$ does contribute to $R$. Then, by lemma 6.2, applied to the derivation $\overline{D}$, there must be a redex family of name $\gamma <^* \alpha$ in $Q$. Furthermore by lemma 4.4, applied to $\overline{C}$ and $\overline{B}$, there exists a redex of name $\delta <^* \gamma$ in $P$, and a redex of name $\eta <^* \delta$ in $M$. By transitivity, $\eta <^* \alpha$. Therefore, by the consistency of $M$, we get $\alpha = \eta$, and then, $\alpha = \delta$. □

**Lemma 6.7** *Let $\overline{A}$ : $M \to^* N$ be a complete-call-by-need derivation, where $M$ is a consistent term. Then, for any decomposition $\overline{A} = \overline{A_1}\,\overline{A_2}$, both derivations $\overline{A_1}$ and $\overline{A_2}$ are complete-call-by-need.*

**Proof**: The result for the prefix derivation $\overline{A_1}$ requires a full proof by induction on $|\overline{A_1}|$.

- Case $|\overline{A_1}| = 1$. By lemma 6.6, applied to $\overline{A}$ and $\overline{A_1}$, any redex family needed by $\overline{A}$ and contracted by $\overline{A_1}$ is also needed by $\overline{A_1}$.

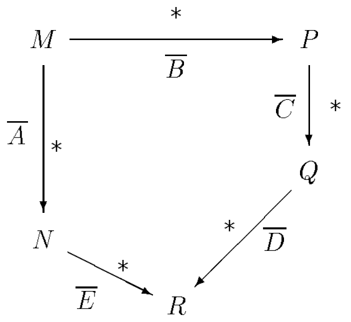


Figure 7: Proof of lemma 6.6

- Case $\overline{A_1} = \overline{B_1}\,\overline{C_1}$. By induction hypothesis, the derivation $\overline{B_1}$ is CCBN. Moreover, the derivation $\overline{C_1}\,\overline{A_2}$ is CCBN as a suffix of $\overline{A}$. Therefore, by induction hypothesis, $\overline{C_1}$ is CCBN. Then $\overline{A_1} = \overline{B_1}\,\overline{C_1}$ is CCBN, since CCBN are closed under concatenation.□

Complete-call-by-need derivations are also closed by residuals :

**Lemma 6.8** *Let $M$ be a consistent term, $\overline{A}$: $M \to^* N$ be a complete-call-by-need derivation, and $\overline{B}$: $M \to^* Q$ be a complete derivation. Then, the derivation $\overline{A}\backslash\overline{B}$ : $P \to^* Q$ is complete-call-by-need.*

**Proof**: By induction on $|\overline{A}|$.

- Assume $|\overline{A}| = 1$. By lemma 6.6, any complete family $\overline{\mathcal{F}}$ contracted by $\overline{A}$ (and therefore needed by $A$ CCBN), is needed by $\overline{B}(\overline{A}\backslash\overline{B})$, since $\overline{\mathcal{F}}$ cannot have any residuals by $\overline{B}(\overline{A}\backslash\overline{B})$. Therefore if $\overline{\mathcal{F}}\backslash\overline{B}$ is non-empty then it is needed by $\overline{A}\backslash\overline{B}$, and the reduction $\overline{A}\backslash\overline{B}$ contracts needed families.
- If $\overline{A} = \overline{A_1}\,\overline{A_2}$, then $\overline{A}\backslash\overline{B} = (\overline{A_1}\backslash\overline{B})(\overline{A_2}\backslash(\overline{B}\backslash\overline{A_1}))$. The induction hypothesis directly applies to $\overline{A_1}$ and $\overline{A_2}$, which are CCBN by lemma 6.7.□

Intuitively, a CCBN derivation never contracts "useless" families. So, the CCBN derivations should be less expensive than other complete derivations,

**Proposition 6.9** *Let $\overline{A}$: $M \to^* N$ be a complete-call-by-need derivation, and $\overline{B}$ : $M \to^* Q$ be a complete derivation, where $M$ is a consistent term. Then, the inequality $\mathcal{C}(\overline{A} \sqcup \overline{B}) \leq \mathcal{C}(\overline{B} \sqcup \overline{A})$ holds.*

**Proof**: By induction on $|\overline{A}|$.

- If $|\overline{A}| = 1$, then let $\overline{\mathcal{F}_1}, \overline{\mathcal{F}_2}, \ldots \overline{\mathcal{F}_n}$ be the complete redex families contracted by $\overline{A}$. Let $\alpha_1, \alpha_2, \ldots \alpha_n$ be their names. By lemma 6.6, the derivation $\overline{B} \sqcup \overline{A}$ *must* contract the families $\overline{\mathcal{F}_1}, \overline{\mathcal{F}_2}, \ldots \overline{\mathcal{F}_n}$. Suppose, for instance, that the derivation $\overline{B}$ contracts the families of names $\alpha_1, \alpha_2 \ldots \alpha_m$ with $m \leq n$, whereas the reduction $\overline{A}\backslash\overline{B}$ contracts the families of names $\alpha_{m+1}, \alpha_{m+2} \ldots \alpha_n$ (by remark 4.1, such a partition is possible). The cost of $\overline{B}$ is at least $m$, therefore there exists an integer $k$ such that $\mathcal{C}(\overline{B}) = m + k$. Whereas the cost of $\overline{A}\backslash\overline{B}$ is exactly $n - m$. Hence, $\mathcal{C}(\overline{B} \sqcup \overline{A}) = n + k$. Now consider the derivation $\overline{A} \sqcup \overline{B}$. The cost of $\overline{A}$ is exactly $n$. The derivation $\overline{B}\backslash\overline{A}$ contract families with the same names as the families contracted by $\overline{B}$. But it does certainly not contract the families of names $\alpha_1, \alpha_2, \ldots \alpha_m$, since these families are already contracted by $\overline{A}$ (remark 4.1, again). Hence $\mathcal{C}(\overline{B}\backslash\overline{A}) \leq k$, and $\mathcal{C}(\overline{A} \sqcup \overline{B}) \leq n + k$.

- If $\overline{A} = \overline{A_1}\,\overline{A_2}$. By the inductive definition of $\backslash$ :
  $\mathcal{C}(\overline{A} \sqcup \overline{B}) = \mathcal{C}(\overline{A_1}) + \mathcal{C}(\overline{A_2}) + \mathcal{C}((\overline{B}\backslash\overline{A_1})\backslash\overline{A_2})$
  $\mathcal{C}(\overline{B} \sqcup \overline{A}) = \mathcal{C}(\overline{B}) + \mathcal{C}(\overline{A_1}\backslash\overline{B}) + \mathcal{C}(\overline{A_2}\backslash(\overline{B}\backslash\overline{A_1}))$

  Or :

  $$\mathcal{C}(\overline{A} \sqcup \overline{B}) = \mathcal{C}(\overline{A_1} \sqcup \overline{B}) + \mathcal{C}(\overline{A_2} \sqcup (\overline{B}\backslash A_1)) - \mathcal{C}(\overline{B}\backslash\overline{A_1})$$
  $$\mathcal{C}(\overline{B} \sqcup \overline{A}) = \mathcal{C}(\overline{B} \sqcup \overline{A_1}) + \mathcal{C}((\overline{B}\backslash\overline{A_1}) \sqcup \overline{A_2}) - \mathcal{C}(\overline{B}\backslash\overline{A_1})$$

  The result follows, since both $\overline{A_1}$ and $\overline{A_2}$ are CCBN.□

This important result demonstrates that it is alway more efficient to start a complete derivation with a call-by-need step than with any other step.

## 6.2 Existence of complete-call-by-need

The CCBN derivations have nice properties, but their definition is not very constructive. In this section, for any complete derivation $\overline{A}$, it is shown that there exists a CCBN derivation $\overline{B}$, such that :

(1) $\overline{B} \sqsubseteq \overline{A}$

(2) $\overline{C}$ CCBN and $\overline{C} \sqsubseteq \overline{A} \;\Rightarrow\; \overline{C} \sqsubseteq \overline{B}$

The derivation $\overline{B}$ is a maximal CCBN derivation included in $\overline{A}$. It will be constructed by induction over $\overline{A}$. A restricted version of the finite development theorem will ensure the termination of this construction :

**Lemma 6.10 (Termination)** *Let $\overline{A} : M \rightarrow^* N$ be a complete derivation. Let $\overline{\mathcal{F}}$ be a complete redex family in $M$, contracted by $\overline{A}$. Let $\overline{F} : M \rightarrow Q$ be the complete reduction, starting from $M$, contracting the family $\overline{\mathcal{F}}$. Then, $\mathcal{C}(\overline{A}\backslash\overline{F}) < \mathcal{C}(\overline{A})$*

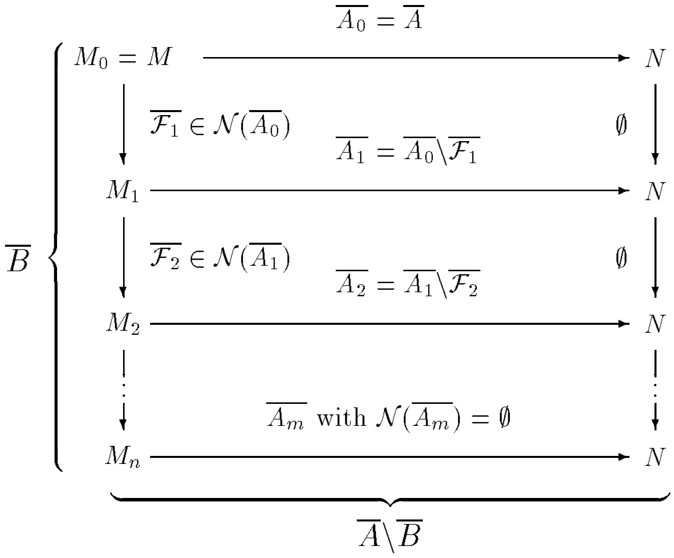



Figure 8: Maximal CCBN $\overline{B}$ included in $\overline{A}$

**Proof**: The derivation $\overline{A}\backslash\overline{F}$ contracts redex families of same names as the redex families contracted by $\overline{A}$. But, it certainly does not contract a family of name $\alpha$. Hence the result. □

Let $M$ be consistent term. Let $\overline{A} : M \rightarrow^* N$ be a complete derivation. We now construct a maximal CCBN derivation $\overline{B}$, included in $\overline{A}$. This is done by the following inductive process (figure 8). If there is no family needed by $\overline{A}$ in $M$, take $\overline{B} = \emptyset$. Otherwise, there exists a complete redex family $\overline{\mathcal{F}}$ in $\mathcal{N}(\overline{A})$. Let $\overline{F} : M \rightarrow M'$ be the complete reduction starting from $M$ and contracting the family $\overline{\mathcal{F}}$. Notice that $\overline{F} \sqsubseteq \overline{A}$, since $\overline{A}$ contracts the family $\overline{\mathcal{F}}$. Take for $\overline{B}$ the concatenation of $\overline{F}$ and $\overline{B'}$, a maximal CCBN derivation included in $\overline{A}\backslash\overline{F}$. By induction hypothesis, $\overline{B'}$ is CCBN. Whereas, by lemma 6.6, $\overline{F}$ is also CCBN. Therefore, $\overline{F}\,\overline{B'}$ is CCBN.

By lemma 6.10 this process terminates, since the positive integer $\mathcal{C}(\overline{A}\backslash\overline{B})$ strictly decreases at each step. In fact, the construction stops when $\mathcal{N}(\overline{A}\backslash\overline{B}) = \emptyset$, which may occur before $\mathcal{C}(\overline{A}\backslash\overline{B}) = 0$. By construction, we get $\overline{B} \sqsubseteq \overline{A}$ (since at each step in the process, $\overline{F}\backslash\overline{A} = \emptyset$). It remains to be proved that $\overline{B}$ is indeed maximal. Let $\overline{C}$ be any CCBN derivation included in $A$. The proposition $\overline{C} \sqsubseteq \overline{A} \Rightarrow \overline{C}\backslash\overline{B} \sqsubseteq \overline{A}\backslash\overline{B}$ is true in the general case and easy to show from the definitions. Therefore, here, we get $\overline{C}\backslash\overline{B} \sqsubseteq \overline{A}\backslash\overline{B}$, and $\mathcal{N}(\overline{C}\backslash\overline{B}) \subseteq \mathcal{N}(\overline{A}\backslash\overline{B}) = \emptyset$. But, $\overline{C}\backslash\overline{B}$ is CCBN, since $\overline{C}$ is CCBN. Therefore $\overline{C}\backslash\overline{B}$, which can only contract families in $\mathcal{N}(\overline{C}\backslash\overline{B})$, is empty, and $\overline{C} \sqsubseteq \overline{B}$.

The construction of a maximal CCBN derivation included in a derivation $A$, is now specialized to the case where $A$ is a normalizing derivation (i.e. $A$ leads to normal form).

**Lemma 6.11** *Let $A\colon M \longrightarrow^* N$ be a non-empty derivation. Then, there exists a redex of name $\alpha$ in $M$, such that, either $\alpha$ is the name of a redex in $N$, or $\alpha$ contributes to $N$ and a residual of $R$ is contracted by $A$.*

**Proof**: From definitions and lemma 4.4. $\square$

**Corollary 6.12** *Let $\overline{A} : M \longrightarrow^* N$ be a non-empty normalizing complete derivation. Then, $\mathcal{N}(\overline{A})$ is not empty.*

**Proof**: Since $\overline{A}$ is non-empty and there is no redex in $N$, there must be a redex family in $M$, contributing to $N$. Furthermore, there is no non-empty derivation starting from the normal form $N$. So, any redex family contributing to $N$ is needed by $\overline{A}$. $\square$

This means that, for any normalizable term, there exists a normalizing CCBN derivation. To prove this, we construct a suitable derivation as follows :

(1) Since $M$ has a normal form $N$, there is a complete normalizing derivation $\overline{A}$ (In the general case, take the Gross derivation, which, at each step, contracts all the available redex families).

(2) Construct a maximal CCBN included in $\overline{A}$. This yields the CCBN derivation $\overline{B}$. By construction, $\overline{B}\backslash\overline{A}$ is normalizing and $\mathcal{N}(\overline{B}\backslash\overline{A}) = \emptyset$. Hence, by corollary 6.12, $\overline{B}\backslash\overline{A}$ is empty and $\overline{B}$ is normalizing.

In fact, this construction does not require the full knowledge of a normalizing derivation. What is really needed, at each step, is a family in $M_i$ which contributes to the normal form. In some particular cases, it is possible to find such a family only by looking at $M_i$.

**Definition 6.13 (Left systems)** *A $T^l RS$ is said to be a left system, iff, for any redex name $\alpha$, and any* allowed *variable occurrence $u$ in $\alpha$, the subterms of $\alpha$ located at occurrences after $u$ in the leftmost-outermost ordering are all variables.*

This notion was defined in [9, 16] for regular TRS, we extend it to $\mathrm{T}^l$RS. It is easy to check that both supercombinators and the weak $\lambda\sigma$-calculus are left systems. This definition has the following important consequence :

**Lemma 6.14** *In a left system, let $M$ be a term. Let $A : M \longrightarrow^* N$ be a derivation. Let $R$ be the leftmost-outermost redex in $M$. Let $\alpha$ be the name of $R$. Let $S$ be the leftmost-outermost redex in $N$ (if it exists), and let $u$ be its occurrence. Then, either $S$ is the only residual of $R$ by $A$, or $\alpha$ contributes to the name of $S$, or $\alpha$ contributes to $N$, in a label at an occurrence before $u$ in the leftmost-outermost ordering.*

**Proof**: Tedious case analysis. $\square$

In a left system, let $M$ be a consistent, normalizable term of normal form $N$. Then, from the previous lemma, when $M$ is not already in normal form, the family of the leftmost-outermost redex in $M$ always contributes to $N$. Therefore, the construction of a normalizing CCBN derivation may yield the complete leftmost-outermost derivation. It suffices to select, at each step, the family of the leftmost-outermost redex, which is known to be needed by any complete normalizing derivation.

In the general case of (unconditional) Orthogonal TRS, the completions of the "standard" or "call-by-name" derivations as defined by Huet and Lévy in [9], are also CCBN derivations. Fully exposing this result would go beyond the scope of this paper.

To summarize, we are able to construct a normalizing CCBN in all cases, and in many practical cases, we showed well known "standard" derivations to be CCBN.

# 7 Optimal normalizing derivations

A result of optimality for reductions leading to normal form follows directly from the preceding sections.

**Theorem 7.1** *Let $M$ be a normalizable consistent term and $N$ its normal form. Let $\overline{A}\colon M \longrightarrow^* N$ be a complete call-by-need derivation from $M$ to $N$. The cost of $\overline{A}$ is lower than the cost of any other normalizing derivation starting from $M$, that is, $\overline{A}$ is optimal.*

**Proof**: Let $B$ be any normalizing derivation $B$. Then, because $B \sqsubseteq \overline{B}$, its completion $\overline{B}$ is also normalizing. Since there is no redex left in the normal form $N$, we have $\overline{A}\backslash\overline{B} = \emptyset^{|A|}$ and $\overline{B}\backslash\overline{A} = \emptyset^{|B|}$ (i.e. $\overline{A}$ and $\overline{B}$ are equivalent). Hence : $\mathcal{C}(\overline{A} \sqcup \overline{B}) = \mathcal{C}(\overline{A})$, and $\mathcal{C}(\overline{B} \sqcup \overline{A}) = \mathcal{C}(\overline{B})$. So, by lemmas 4.14 and 6.9, we get : $\mathcal{C}(B) \geq \mathcal{C}(\overline{B}) \geq \mathcal{C}(\overline{A})$. $\square$

This proves the following intuitive result : a derivation contracting only redexes that have to be contracted, while avoiding duplication, is optimal for a cost measure taking sharing into account. Our notion of sharing is sensible, since it corresponds exactly to ordinary acyclic graphs, as used by actual graph reducers.

Therefore, for the supercombinators as well as for the weak $\lambda\sigma$-calculus, the leftmost-outermost strategy with sharing is an optimal one. In the case of the weak $\lambda\sigma$-calculus, the substitution steps are taken into account, making our cost measure close to implementation cost.

## Conclusion

The new notions of conditional Term Rewriting Systems and labeled Term Rewriting Systems have been introduced, their addition to TRS yields the more general $T^l$RS framework. This new class of reduction systems deserves a full study in itself. For the Orthogonal $T^l$RS, a large significant subclass of $T^l$RS, there is an optimal derivation leading to normal form. Furthermore, this optimal derivation may be implemented with ordinary graphs. This paper extends the optimality result already known for the recursive program schemes [3, 19] and the graphical SK combinators [12] to Orthogonal $T^l$RS, a clear, convenient and more abstract formalism.

In all cases, the optimal normalizing derivation may be effectively computed from any normalizing derivation. In many practical cases, the optimal derivation may be computed from the starting term only. In particular, we prove the leftmost-outermost strategy with sharing to be optimal in two distinct forms of weak $\lambda$-calculi, formalizing a result widely known as a "folk's theorem". The exact proof is complicated by the necessity of extending the TRS formalism in two new directions. As demonstrated in [5], finding optimal derivations in the full $\lambda$-calculus is a much harder problem, and the effective strategies proposed by [11, 13] along the theoretical work of [15] are more complicated than simple lazy strategy.

## Acknowledgments

I thank Jean-Jacques Lévy for many helpful discussions. I also thank John Bradley Chen and Xavier Leroy for their editorial work.

## References

[1] M. Abadi, L. Cardelli, P.-L. Curien, J.-J. Lévy, "Explicit substitutions". POPL'90.

[2] L. Augustsson, "Compiling Lazy Functional Languages Part II". PhD thesis, Chalmers University of Technology, Sweden, 1987.

[3] G. Berry, J.-J. Lévy, "Minimal and Optimal Computations of Recursive Programs". JACM Vol. 26, No 1, 1979.

[4] G. Cousineau, P.-L. Curien, M. Mauny, "The Categorical Abstract Machine". Functional Programming Languages an Computer Architecture, 1985, LCNS.

[5] J. Field, "On Laziness and Optimality in Lambda Interpreters : Tools for Specification and Analysis". POPL'90.

[6] T. Hardin, J.-J. Lévy, "A confluent Calculus of Substitution". France-Japan Artificial Inteligence and Computer Science Symposium, Izu, 1989.

[7] J.R. Hindley, J.P. Seldin, "Introduction to Combinators and $\lambda$-calculus". London Mathematical Society, Students Texts 1, Cambridge University Press, 1986.

[8] B.T. Howard, J.C. Mitchell, "Operational and Axiomatic Semantics of PCF". L&FP'90.

[9] G. Huet, J.-J. Lévy, "Call by Need Computations in Non-Ambiguous Linear Term Rewriting Systems". INRIA, technical report 359, 1979.

[10] T. Johnsson, "Compiling Lazy Functional Languages". PhD thesis, Chalmers University of Technology, Sweden, 1987.

[11] V. K. Kathail, "Optimal Interpreters for Lambda-calculus Based Functional Languages". Communication at the SEMAGRAPH workshop, Paris, April 1990.

[12] J.R. Kennaway, "An Outline of Some Results of Staples on Optimal Reduction Orders in Replacements Systems". Internal Report CSA/19/1984.

[13] J. Lamping, "An Algorithm for Optimal Lambda Calculus Reduction". POPL'90.

[14] X. Leroy, "The Zinc experiment : an economical implementation of the ML Language". INRIA, technical report 117, 1989.

[15] J.-J. Lévy, "Réductions correctes et optimales dans le $\lambda$-calcul". Thèse de Doctorat d'Etat, Université Paris VII, 1978.

[16] M. O'Donnell, "Computing in Systems Described by Equations". LNCS No 58, Eds G. Goos and J. Hartmanis, Springer, 1977.

[17] S. L. Peyton Jones, "The Implementation of Functional Programming Languages". Prentice-Hall, 1987.

[18] G.D. Plotkin, "LCF Considered as a Programming Language". Theoretical Computer Science, 5:225-255, 1977.

[19] J. Vuillemin, "Syntaxe, Sémantique et axiomatique d'un language de programmation simple". Thèse de doctorat d'état, Université Paris VI, 1974.