Tracking Redexes in the Lambda Calculus

Jean-Jacques Lévy

Abstract Residuals of redexes keep track of redexes along reductions in the lambda calculus. Families of redexes keep track of redexes created along these reductions. In this paper, we review these notions and their relation to a labeled λ-calculus introduced here in a systematic way. These properties may be extended to combinatory logic, term rewriting systems, process calculi and proofnets of linear logic.

1 Introduction

The λ-calculus is a basic setting in the foundations of mathematical logic. It gained much importance in proof theory within types and the Curry-Howard correspondence. In the theory of programming languages, the λ-calculus is a key model for functional languages. It has also applications in their type theory and even in imperative languages through the use of continuations. The beauty of the λ-calculus is that all calculations, named reductions, are generated by a single rule, the β-conversion rule, which define the application of an argument to a function.

We suppose that the reader is familiar with the usual definitions and notations of the λ-calculus. If not the reader is refered to Barendregt's book [6]. We will mainly consider β-conversion and β-redexes, although many of the results exposed here also hold in other calculi. A reductible expression (redex) is any term of the form $(\lambda x.M)N$ where argument N is applied to the function $\lambda x.M$. Its contraction produces the term $M\{x := N\}$ in which all occurrences of the free variable x in M are replaced by N. We ignore all problems due to the renaming of bound variables (α-conversion) and assume that the binding of bound variables are respected as in standard mathematics.

Although β-conversion is a simple rule, many results of the λ-calculus are not easy to prove. This is due to the ability of computing inside the body of functions

Jean-Jacques Lévy
Irif & Inria Paris e-mail: jean-jacques.levy@inria.fr

at the same time as their arguments are passed to functions. Moreover reductions may be infinite and inductive proofs have to be carefully performed. However in the typed versions of the λ-calculus, there are no infinite reductions and proofs get much simpler. In this paper, we show how the untyped λ-calculus can be viewed as an infinite limit of labeled calculi, and how proofs in the untyped case can be conducted in these pseudo-typed calculi.

The results in this paper are already present in many old articles [21, 23], but our presentation is focused here on a systematic way of defining a labeled calculus that we will show related to the notion of family of redexes. This family relation generalizes the notion of residuals which appeared in Church original monograph [11]. We start from the Hyland-Wadsworth calculus in section 2 and derive the labeled calculus. In section 3, we show its correspondence with permutation equivalence. The history of redexes is related to labeled redexes in section 4. We explain our the results are applicable to combinatory logic and orthogonal term rewriting systems in section 5. In the conclusion, we cite several related topics such as optimal reductions, reversible calculi, causality and event structures.

2 The labeled lambda-calculus

Hyland and Wadsworth introduced an indexed λ-calculus as a syntactic model for Scott D_∞ model [30, 33]. This calculus can be considered as a typed approximation of the untyped λ-calculus[22], since every reduction in this calculus is finite and the normal forms are unique. Both the Church-Rosser theorem and strong normalization are valid in Hyland-Wadsworth λ-calculus, which is more closely defined as follows:

$$M, N \;::=\; x^n \mid (\lambda x.M)^n \mid (M\,N)^n \qquad (n \text{ is any natural number})$$

$$((\lambda x.M)^{n+1} N)^m \;\rightarrow\; M\{x := N_{[n]}\}_{[n]\,[m]}$$

$$\begin{aligned} x^n\{x := P\} &= P_{[n]} & x^n_{[m]} &= x^{\inf\{n,m\}} \\ (\lambda y.M)^n\{x := P\} &= (\lambda y.M\{x := P\})^n & (\lambda y.M)^n_{[m]} &= (\lambda y.M)^{\inf\{n,m\}} \\ (M\,N)^n\{x := P\} &= (M\{x := P\}\,N\{x := P\})^n & (M\,N)^n_{[m]} &= (M\,N)^{\inf\{n,m\}} \end{aligned}$$

Notice that redex $((\lambda x.M)^n N)^m$ can be contracted only when $n > 0$. Therefore not every reduction of the untyped λ-calculus can be simulated in this indexed λ-calculus. For instance, let $\Delta_n = (\lambda x.(x^{10} x^4)^{20})^n$, then:

$$(\Delta_3\,\Delta_4)^{15} \rightarrow (\Delta_2\,\Delta_2)^2 \rightarrow (\Delta_1\,\Delta_1)^1 \rightarrow (\Delta_0\,\Delta_0)^0$$

whereas in the untyped λ-calculus, the term $(\lambda x.x\,x)(\lambda x.x\,x)$ loops and therefore does not normalize. Moreover the indexed calculus enjoys the Church-Rosser property. The confluency is visible on the example of figure 1. The calculation of exponents may look complex, but it consists in alternating applications of the predecessor function and of the minimum of two natural numbers. Indeed redexes have a distinct

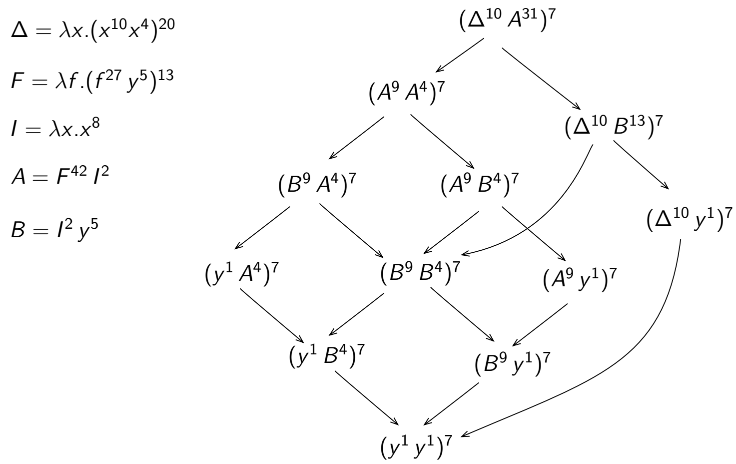

Fig. 1 A reduction graph in Hyland-Wadsworth λ-calculus

computing power depending upon the exponent of their function part, namely the exponent n in $((\lambda x.M)^n\,N)^m$. When $n = 0$, the redex is frozen. When n is large, the redex can compute as in the usual untyped calculus. In our example, let $\Omega_3 = (\Delta_3\Delta_4)^{15}$ and $\Omega_2 = (\Delta_2\Delta_2)^2$, then Ω_3 can perform 3 steps of reduction, but Ω_2 can perform 2 steps. The exponent of the function part of a redex is named the degree of the redex. Thus the degree 3 of Ω_3 is larger than the degree 2 of Ω_2.

The proof of the confluency in the indexed calculus can be found in [21]. (Thanks to G.Plotkin who taught me the Tait-Martin-Löf method) The proof of strong normalization can be found in [23] (Thanks to D.van Daalen who taught me his easy way of proving it). Thus the indexed calculus is confluent and only performs finite reductions. Each term has a unique normal form. There is also the simulation property, which means that any finite reduction of the untyped calculus may be emulated in the indexed calculus as soon as one gives sufficiently large exponents in the initial term. Therefore the reduction graphs of the untyped λ-calculus may be seen as an infinite limit of compact reduction graphs, namely confluent reductions of finite graphs. These graphs corresponds to the inverse-limit construction of the D_∞ model, but they also have an intensional meaning about reductions in the untyped λ-calculus.

In figure 1, all reductions of untyped λ-calculus are emulated. But the reduction graph differs when the exponents are lowered in the initial term as shown by figure 2. These compactness and confluency properties of reduction graphs was used in [21] to show the inside-out completeness of reductions in the untyped λ-calculus. There are other ways of proving this completeness, but the proof with the indexed calculus is notoriously simpler and quite intuitive, since innermost reductions lead to the normal forms when strong normalization is present.

When all exponents of a term are null, the term is in normal form, since there is no way of contracting a 0-degree redex. When all exponents are 1 in the initial term, its redexes are activated, but only these ones. The new redexes which may appear by applying a function to a function argument or by de-curryfication are frozen. When

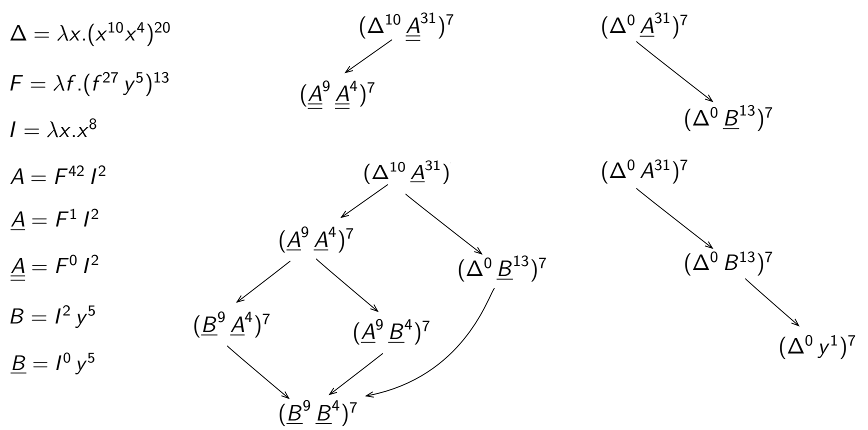

Fig. 2 By varying exponents in the initial term, these graphs approximate the reduction graph of figure 1. Notice that they are all finite and confluent.

all exponents of the initial term are 2, the situation is less clear. The new redexes are activated, but no more. See for instance the redexes B and $\underline{B}$ in figure 2.

Another interesting property of the indexed λ-calculus is that the residuals of redexes keep their degree. Residuals are defined in the Curry&Feys volume [13]. Intuitively residuals are the copies or the remainings of redexes when another redex is contracted. Therefore, redexes keep their computing power when other redexes are computed. So if one restricts the contraction of a redex when a predicate is valid on the degree of this redex, the calculus is still confluent. For instance, one contracts $((\lambda x.M)^n N)^m$ if and only if n is odd.

Finally the Hyland-Wasworth indexed calculus is a bit frustrating, since it looks as a particular case of a more general property which makes the compactness and the confluency properties of the untyped λ-calculus. In the appendix of [23], a systematic way of generalizating this calculus is exhibited, but it was not much popularized. In this short note, we will show it, 45 years later!

We are therefore searching for a calculus where every subterm has an abstract label as an exponent and a reduction rule which preserves confluency and preservation of the degrees for redexes. Moreover we look for a simple condition on degrees of redexes in order to get strong normalization. This calculus may look as follows:

$$M, N \;::=\; x^\alpha \;\mid\; (\lambda x.M)^\alpha \;\mid\; (M\,N)^\alpha \qquad (\alpha, \beta, \ldots \text{ are abstract labels})$$

$$((\lambda x.M)^\alpha N)^\beta \;\rightarrow\; h(\alpha, \beta, M\{x := g(\alpha, \beta, N)\})$$

$$x^\alpha\{x := P\} = f(\alpha, P)$$
$$(\lambda y.M)^\alpha\{x := P\} = (\lambda y.N\{x := P\})^\alpha$$
$$(M\,N)^\alpha\{x := P\} = (M\{x := P\}\,N\{x := P\})^\alpha$$

where f, g, h are undefined functions producing a new term of this calculus. We used the inductive definition of substitution which keeps invariant the labels except for variables where the f function is applied. Thus we preserve the degrees of redexes inside the term which is substituted. We also copy Hyland-Wadsworth calculus by giving local effects to f, g, h on the external exponents of terms. This again keeps invariant the degrees of redexes.

$$h(\alpha,\beta,x^\gamma) = x^{\psi(\alpha,\beta,\gamma)} \qquad g(\alpha,\beta,x^\gamma) = x^{\chi(\alpha,\beta,\gamma)}$$
$$h(\alpha,\beta,(\lambda y.M)^\gamma) = (\lambda y.M)^{\psi(\alpha,\beta,\gamma)} \qquad g(\alpha,\beta,(\lambda y.M)^\gamma) = (\lambda y.M)^{\chi(\alpha,\beta,\gamma)}$$
$$h(\alpha,\beta,(M\,N)^\gamma) = (M\,N)^{\psi(\alpha,\beta,\gamma)} \qquad g(\alpha,\beta,(M\,N)^\gamma) = (M\,N)^{\chi(\alpha,\beta,\gamma)}$$

$$f(\alpha,x^\beta) = x^{\phi(\alpha,\beta)}$$
$$f(\alpha,(\lambda y.M)^\beta) = (\lambda y.M)^{\phi(\alpha,\beta})$$
$$f(\alpha,(M\,N)^\beta) = (M\,N)^{\phi(\alpha,\beta)}$$

where ϕ,χ,ψ are undefined functions used to define f,g,h. We now try to prove local confluency of this calculus, i.e. the permutation of the contraction of two redexes in any given term M.

Goal 1 If $M \to N$ and $M \to P$, there exists Q such that $N \twoheadrightarrow Q$ and $P \twoheadrightarrow Q$.

Proof : By induction on the size $\|M\|$ of M. The only interesting case is when $M = ((\lambda x.M_1)^\alpha M_2)^\beta$ and $N = h(\alpha,\beta,N_1\{x := g(\alpha,\beta,N_2)\})$ and we have $P = ((\lambda x.P_1)^\alpha P_2)^\beta$ with $M_1 \to P_1$ and $M_2 = P_2$ or $M_1 = P_1$ and $M_2 \to P_2$. The next goal is to show $h(\alpha,\beta,N_1\{x := g(\alpha,\beta,N_2)\}) \twoheadrightarrow h(\alpha,\beta,P_1\{x := g(\alpha,\beta,P_2)\})$. We subdivide this goal into the following 4 subgoals.

Goal 2 If $N \to P$, then $M\{x := N\} \twoheadrightarrow M\{x := P\}$.

Goal 3 If $M \to N$, then $M\{x := P\} \twoheadrightarrow N\{x := P\}$.

Goal 4 If $M \to N$, then $h(\alpha,\beta,M) \twoheadrightarrow h(\alpha,\beta,N)$.

Goal 5 If $M \to N$, then $g(\alpha,\beta,M) \twoheadrightarrow g(\alpha,\beta,N)$.

Proof (G2) : by induction on $\|M\|$. When $M = x^\alpha$, one has to prove:

Goal 6 If $M \to N$, then $f(\alpha,M) \twoheadrightarrow f(\alpha,N)$.

Proof (G3) : By induction on $\|M\|$. The critical case is when $M = ((\lambda y.M_1)^\alpha M_2)^\beta \to N = h(\alpha,\beta,M_1\{y := g(\alpha,\beta,M_2)\})$. Then $M\{x := P\} = ((\lambda y.M_1')^\alpha M_2')^\beta$ with $M_1' = M_1\{x := P\}$ and $M_2' = M_2\{x := P\}$. One has to show:

$$h(\alpha,\beta,M_1\{y := g(\alpha,\beta,M_2)\}\{x := P\} = h(\alpha,\beta,M_1'\{y := g(\alpha,\beta,M_2')\})$$

which we subdivide into 3 new goals.

Goal 7 $M\{x := N\}\{y := P\} = M\{y := P\}\{x := \{y := P\}\}$ when x is not free in P.

Goal 8 $g(\alpha,\beta,M)\{x := N\} = g(\alpha,\beta,M\{x := N\})$

Goal 9 $h(\alpha,\beta,M)\{x := N\} = h(\alpha,\beta,M\{x := N\})$

Proof (G4) : Obvious except when $M = ((\lambda x.M_1)^\gamma M_2)^\delta \to N = h(\gamma,\delta,M_1\{x := g(\gamma,\delta,M_2)\})$. The definition of h gives $h(\alpha,\beta,M) = ((\lambda x.M_1)^\gamma M_2)^\epsilon$ with $\epsilon = \psi(\alpha,\beta,\delta)$. Therefore $h(\alpha,\beta,M) \to N' = h(\gamma,\epsilon,M_1\{x := g(\gamma,\epsilon,M_2)\})$. It suffices to show $N' = h(\alpha,\beta,N)$. This leads to satify the following 2 equations on the algebra of labels.

(E 1) $\psi(\alpha,\beta,\psi(\gamma,\delta,\epsilon)) = \psi(\gamma,\psi(\alpha,\beta,\delta),\epsilon)$

(E 2) $\chi(\gamma,\delta,\epsilon) = \chi(\gamma,\psi(\alpha,\beta,\delta),\epsilon)$

Proof (G5) : As previously. The critical case leads to equations for g in place of h.

(E 3) $\chi(\alpha,\beta,\psi(\gamma,\delta,\epsilon)) = \psi(\gamma,\chi(\alpha,\beta,\delta),\epsilon)$

(E 4) $\chi(\gamma,\delta,\epsilon) = \chi(\gamma,\chi(\alpha,\beta,\delta),\epsilon)$

Proof (G6) : Again when the contracted redex is at toplevel, we get:

(E 5) $\phi(\alpha,\psi(\beta,\gamma,\delta)) = \psi(\beta,\phi(\alpha,\gamma),\delta)$

(E 6) $\chi(\beta,\gamma,\delta) = \chi(\beta,\phi(\alpha,\gamma),\delta)$

Proof (G7) : By induction on $\|M\|$. This goal is the standard substitution lemma which makes confluency. With labels we have a new critical case when $M = x^\alpha$ producing a new goal.

Goal 10 $f(\alpha,N)\{y := P\} = f(\alpha,N\{y := P\})$

Proof (G8, G9, G10) : By case inspection on the external label, we get 3 more equations to satisfy on labels.

(E 7) $\phi(\chi(\alpha,\beta,\gamma),\delta) = \chi(\alpha,\beta,\phi(\gamma,\delta))$

(E 8) $\phi(\psi(\alpha,\beta,\gamma),\delta) = \psi(\alpha,\beta,\phi(\gamma,\delta))$

(E 9) $\phi(\phi(\alpha,\beta),\gamma) = \phi(\alpha,\phi(\beta,\gamma))$

The solution of these 9 equations will solve the local confluency statement of goal G1. It would have been better to prove the full confluency of this new calculus, for instance with the axiomatic Tait-Martin Löf method. But the same equations are then sufficient. In fact, these equations were firstly obtained by trying to prove the full confluency result. Anyhow, the 9 equations prove local confluency and full conflency may be proved once the labeled calculus is precisely defined.

Equation 9 suggests that ϕ is associative. Therefore we may take α and β as strings in the free monoid and write:

$$\phi(\alpha,\beta) = \alpha\beta$$

Equations 5-8 and 6-7 give following recursive definitions for ψ and χ.

$$\psi(\beta,\alpha\gamma,\delta) = \alpha\psi(\beta,\gamma,\delta) \qquad \chi(\beta,\alpha\gamma,\delta) = \chi(\beta,\gamma,\delta)$$
$$\psi(\alpha,\beta,\gamma\delta) = \psi(\alpha,\beta,\gamma)\delta \qquad \chi(\alpha,\beta,\gamma\delta) = \chi(\alpha,\beta,\gamma)\delta$$

Let o be the empty string and pose $\lceil\alpha\rceil = \psi(\alpha,o,o)$ and $\lfloor\alpha\rfloor = \chi(\alpha,o,o)$. Then:

$$\psi(\alpha,\beta,\gamma) = \beta\lceil\alpha\rceil\gamma \qquad \chi(\alpha,\beta,\gamma) = \lfloor\alpha\rfloor\gamma$$

The labeled lambda-calculus is now fully defined.

$$M,N \ ::= \ x^{\alpha} \ \mid \ (\lambda x.M)^{\alpha} \ \mid \ (M\,N)^{\alpha}$$
$$((\lambda x.M)^{\alpha}N)^{\beta} \ \to \ \beta\cdot\lceil\alpha\rceil\cdot M\{x := \lfloor\alpha\rfloor\cdot N\}$$
$$x^{\alpha}\{x := P\} = \alpha\cdot P \qquad \alpha\cdot x^{\beta} = x^{\alpha\beta}$$
$$(\lambda y.M)^{\alpha}\{x := P\} = (\lambda y.M\{x := P\})^{\alpha} \qquad \alpha\cdot(\lambda y.M)^{\beta} = (\lambda y.M)^{\alpha\beta}$$
$$(M\,N)^{\alpha}\{x := P\} = (M\{x := P\}\,N\{x := P\})^{\alpha} \qquad \alpha\cdot(M\,N)^{\beta} = (M\,N)^{\alpha\beta}$$

We say that label $\lceil\alpha\rceil$ is overlined and $\lfloor\alpha\rfloor$ is underlined. Labels may be atomic letters on any given alphabet or composite strings formed by atomic letters and overlined or underlined labels. These underlined/overlined labels may also be considered as atomic since the calculus cannot break them. They are just a way of building new structured labels for redexes and subterms.

In [22, 23], the labeled λ-calculus is defined as previously. However we notice that the key property for confluency is the associativity of the ϕ function in the above calculations. So we could have defined $\phi(\alpha,\beta) = \beta\alpha$ without breaking associativity. This gives the following definitions where labels are mirrored with respect to the labeled λ-calculus in [22, 23].

$$\phi(\alpha,\beta) = \beta\alpha \qquad \psi(\alpha,\beta,\gamma) = \gamma\lceil\alpha\rceil\beta \qquad \chi(\alpha,\beta,\gamma) = \gamma\lfloor\alpha\rfloor$$

The corresponding labeled λ-calculus matches the structure of terms and contexts in a better way. Moreover we may also allow empty labels. Then the definition of substitution is the standard definition for substitution in the unlabeled λ-calculus. Therefore the labeled λ-calculus is now expressed in the following way. (Let o be the empty label and pose $\lceil o\rceil = \lfloor o\rfloor = o$)

$$M,N \ ::= \ x \ \mid \ \lambda x.M \ \mid \ M\,N \ \mid \ M^{\alpha}$$
$$(\lambda x.M)^{\alpha}N \ \to \ M\{x := N^{\lfloor\alpha\rfloor}\}^{\lceil\alpha\rceil}$$
$$M^{\alpha}\{x := P\} = M\{x := P\}^{\alpha} \qquad (M^{\alpha})^{\beta} = M^{\alpha\beta}$$

This simple calculus seems the most general confluent labeled calculus as soon as we assume local modifications for labels. For instance Hyland-Wadsworth calculus can be obtained by homomorphism on labels by taking $\alpha\beta = \inf\{\alpha,\beta\}$ for the concatenation and $\lceil\alpha\rceil = \lfloor\alpha\rfloor = \alpha - 1$ for overlining and underlining. Moreover the

$\Delta = \lambda x.(x^c x^d)^b$

$F = \lambda f.(f^k\, y^\ell)^j$

$I = \lambda x.x^v$

$A = (F^i\, I^u)^q$

$B = (I^\gamma\, y^\ell)^q$

$C = y^{\ell\lfloor\gamma\rfloor v\lceil\gamma\rceil q}$

$\gamma = u\lfloor i \rfloor k$

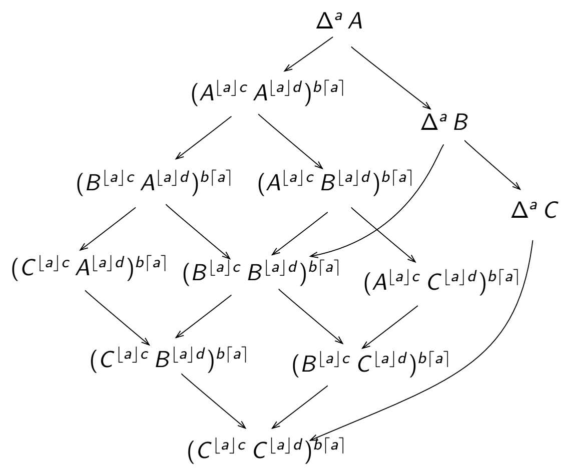

Fig. 3 A labeled reduction graph corresponding to the reduction graph in figure 1

calculus preserves the degree of redexes. With labels instead of numbers, it is more suitable to call name of a redex the label of its function part, what was named the degree of a redex in Hyland-Wadsworth calculus.

Consider now the following example where $\underline{\Delta} = \lambda x.(x^c\, x^d)^b$, $\Delta = \lambda x.(x^g\, x^h)^f$. Then the names of redexes become larger and larger in the same way as their degrees were decreasing in Hyland-Wadsworth calculus.

$$\begin{array}{lll}
\Omega = \underline{\Delta}^a\, \Delta^e & & \\
\quad\to \Omega_1 = (\Delta^{\gamma_1}\, \Delta^{\delta_1})^{b\lceil a\rceil} & \gamma_1 = e\lfloor a\rfloor c & \delta_1 = e\lfloor a\rfloor d \\
\quad\to \Omega_2 = (\Delta^{\gamma_2}\, \Delta^{\delta_2})^{f\lceil\gamma_1\rceil b\lceil a\rceil} & \gamma_2 = \delta_1\lfloor\gamma_1\rfloor g & \delta_2 = \delta_1\lfloor\gamma_1\rfloor h \\
\quad\to \Omega_3 = (\Delta^{\gamma_3}\, \Delta^{\delta_3})^{f\lceil\gamma_2\rceil f\lceil\gamma_1\rceil b\lceil a\rceil} & \gamma_3 = \delta_2\lfloor\gamma_2\rfloor g & \delta_3 = \delta_2\lfloor\gamma_2\rfloor h \\
\quad\to \cdots & &
\end{array}$$

On the example of figure 3, we may check confluency. Moreover there are three redexes with names a, i and $\gamma = u\lfloor i\rfloor k$. The first two ones are in the initial term, but the third one is created by the contraction of the redex with name i. Its name is made of the name of his creator and the labels of two other subterms in the context. Similarly in the above reduction of Ω, there are redexes with names a, γ_1, $\gamma_2, \gamma_3, \ldots$ and the names of their creators are also part of the names of these redexes. Sometimes two redexes must be contracted to create a third one. Take for instance $F = \lambda f.(f^c\, y^d)^b$, $I = \lambda x.x^v$ and $\Delta = \lambda x.(x^k\, x^\ell)^j$. Then the redexes with names a and u are both creators of the redex in the final term of the reduction below. (Notice that the two redexes of the initial term can be contracted in any order and thanks to the associativity of concatenation on labels, the name of the final redex is indeed unique)

$$F^a(I^u\, \Delta^i)^q \to F^a \Delta^{i\lfloor u\rfloor v\lceil u\rceil q} \to (\Delta^{i\lfloor u\rfloor v\lceil u\rceil q\lfloor a\rfloor c}\, y^d)^{b\lceil a\rceil}$$

Our final remark is about strong normalization. The labeled calculus preserves the names of redexes. Technically redexes and their residuals have a same name.

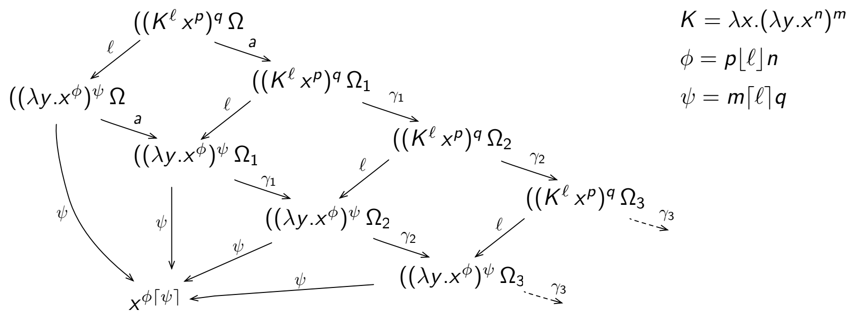

Fig. 4 The reduction graph from $(K^\ell x^p)^q\,\Omega$ with names of contracted redexes on top of arrows.

Therefore it still enjoys the Church-Rosser property when we restrict the conversion rule when a given predicate $\mathcal{P}$ is valid on the names of the contracted redexes. The conversion rule is then

$$(\lambda x.M)^\alpha N \;\rightarrow\; M\{x := N^{\lfloor\alpha\rfloor}\}^{\lceil\alpha\rceil} \qquad \text{when } \mathcal{P}(\alpha) \text{ is true}$$

Then we recover the reduction graphs of figure 2 by considering predicates such that $\mathcal{P}_1(\alpha) \equiv \alpha = a$, $\mathcal{P}_2(\alpha) \equiv \alpha = i$, $\mathcal{P}_3(\alpha) \equiv \alpha \in \{a, i\}$ . . . . A key property is that the labeled λ-calculus strongly normalizes when the predicate $\mathcal{P}$ is only valid on a finite set [23]. Then we cannot have an infinite reduction and the normal form is unique. For instance, on figure 4, take $\Omega = \underline{\Delta}^a\,\Delta^e$ as defined previously, the reduction graph becomes finite when $\mathcal{P}$ is restricted to finite subsets of the set $\{a, \ell, \phi, \psi, \gamma_1, \gamma_2, \cdots\}$ of redex names.

We now summarize the previous definitions and the main results of the labeled λ-calculus.

Definition 1 Let $\mathcal{A} = \{a, b, c, \cdots\}$ be a given finite alphabet of letters. The set of labels $\mathcal{A}^*$ is a set of strings defined as follows

$$\alpha, \beta \;::=\; a \mid \alpha\beta \mid \lceil\alpha\rceil \mid \lfloor\alpha\rfloor \mid o$$

where $\lceil\alpha\rceil$ and $\lfloor\alpha\rfloor$ are overlined / underlined strings and o is the empty label (the empty string) with posing $\lceil o\rceil = \lfloor o\rfloor = o$. Let $\mathcal{A}^+$ is the set of nonempty labels.

Definition 2 Let $\mathcal{P}$ be a predicate on $\mathcal{A}^*$. Then the labeled λ-calculus is defined as:

$$M, N \;::=\; x \mid \lambda x.M \mid M\,N \mid M^\alpha$$

$$(\lambda x.M)^\alpha N \;\rightarrow\; M\{x := N^{\lfloor\alpha\rfloor}\}^{\lceil\alpha\rceil} \qquad \text{when } \mathcal{P}(\alpha) \text{ is true}$$

$$M^\alpha\{x := P\} = M\{x := P\}^\alpha \qquad (M^\alpha)^\beta = M^{\alpha\beta}$$

where the substitution $M\{x := N\}$ is defined as in the standard λ-calculus.

The pure labeled λ-calculus is when every label is nonempty. Notice that when the initial term is pure (i.e. with nonempty labels), then all reducts are pure.

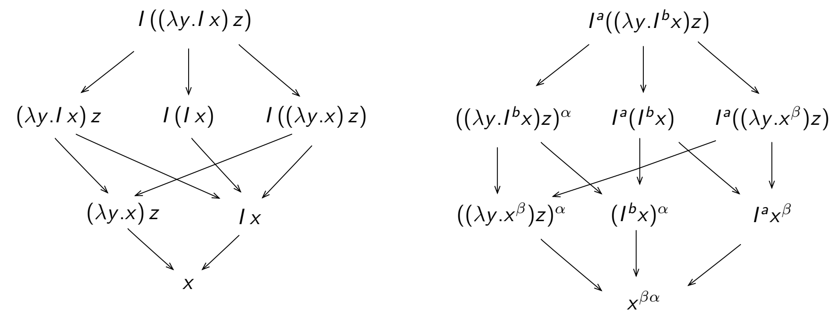

Fig. 5 The reduction graph on left does not form a lattice, but the labeled version on right forms a lattice. Take $I = \lambda x.x$, $\alpha = \lfloor a \rfloor \lceil a \rceil$, $\beta = \lfloor b \rfloor \lceil b \rceil$.

Theorem 1 (confluency) *The labeled λ-calculus is confluent.*

Theorem 2 (strong normalization) *If the domain of $\mathcal{P}$ is finite, the pure labeled λ-calculus strongly normalizes.*

There are many other properties of the labeled calculus. Here, we mention several without their proofs which can be found in [23]. We refer to Curry&Feys book [13] for the exact definition of residuals and standard reductions (i.e. reductions with a left-to-right strategy). A created redex is a redex which is not a residual of another redex [21].

Proposition 1 (preservation of redex names) *In the labeled λ-calculus, any redex and its residuals have the same name.*

Proposition 2 (creation of redexes) *When $M \rightarrow N$ by contracting a redex R in M, the name of R is overlined or underlined in the name of any created redex S in N.*

Theorem 3 (standardization) *In the pure labeled λ-calculus, if $M \twoheadrightarrow N$, then there is a unique standard reduction $M \overset{st}{\twoheadrightarrow} N$ from M to N.*

Proposition 3 (lattice of reductions) *In the pure labeled λ-calculus, the reduction graph has an upper semi-lattice structure.*

Notice that this property is not true in the usual λ-calculus, but indeed holds as shown on figure 5. One can prove that reduction graphs are full-fledged lattices in the λI-calculus where K terms are not allowed [23].

3 Labels and permutation equivalence

Let now consider the standard unlabeled λ-calculus. Reductions are designated by the greek letters ρ, σ, τ. The reduction ρ from M to N contracting the redexes R_1 in M_0 producing M_1, the redex R_2 in M_1 producing M_2, ... the redex R_n in M_{n-1} producing M_n where $n \geq 0$ is written:

$$\rho : M = M_0 \xrightarrow{R_1} M_1 \xrightarrow{R_2} M_2 \cdots \xrightarrow{R_n} M_n = N$$

We write R/ρ for the set of residuals in N of a redex R in M along reduction ρ. Intuitively residuals of a redex are what remains of that redex in the final term of the reduction. The residuals of a redex R in a term M after the contraction of an other redex S are defined by case inspection on the relative positions of R and S in M. There are 6 cases! We illustrate the definition of residuals in the examples below. Let $I = \lambda x.x$ and $\Delta = \lambda x.x\,x$, then the redex R and its residuals R/S are underlined:

$$\begin{array}{lll}
\underline{(I\,x)}\,(I\,x) \;\rightarrow\; \underline{(I\,x)}\,x & \qquad & I\,x\,\underline{(I\,x)} \;\rightarrow\; x\,\underline{(I\,x)} \\
\Delta\,\underline{(I\,x)} \;\rightarrow\; \underline{(I\,x)}\,\underline{(I\,x)} & \qquad & ((\lambda x.(I\,x)\underline{(I\,x)})\,y \;\rightarrow\; I\,y\,\underline{(I\,y)} \\
\underline{\Delta\,(I\,x)} \;\rightarrow\; \underline{(\Delta\,x)} & \qquad & \underline{(I\,x)} \;\rightarrow\; x
\end{array}$$

Notice that when there are more than one residual of R, these residuals are all disjoint. This is the case for a single reduction step, but is no longer true after several steps. Thus when R and S are two redexes in a same term, the residuals R/S and S/R are sets of disjoint redexes.

When R and S are two disjoint redexes in M, we can contract them in any order and obviously obtain the same term. This is also true for any given set $\mathcal{F}$ of disjoint redexes in M. Their contractions give a same term N, which we write:

$$M \xrightarrow{\mathcal{F}} N$$

We write $\rho : \mathcal{F}$ when ρ is a any reduction contracting the disjoint redexes of $\mathcal{F}$ in any order. Let also write $\rho : R$ when $\mathcal{F} = \{R\}$.

When a redex R contains a redex S (or S contains R) in M, we have to consider more carefully the set R/S and S/R of residuals. By the following local-confluency lemma, we get again the same term by contracting R and the disjoint set of residuals S/R in any order or contracting S and the residual R/S. The proof is in [13, 23].

Lemma 1 (basic permutation) *Let R and S be redexes in M. Then if $M \xrightarrow{R} N$ and $M \xrightarrow{S} P$, there exists Q such that $N \xrightarrow{S/R} Q$ and $P \xrightarrow{R/S} Q$.*

Let $\rho\,;\sigma$ be the reduction ρ followed by reduction σ. Let ρ and σ be coinitial when they start from the same term, and cofinal when they end on the same term. Then the permutation equivalence on reductions is defined as follows.

Definition 3 (permutation equivalence) Two coinitial reductions ρ and σ are equivalent by permutations, written $\rho \sim \sigma$, in the following ways:

(i) $\rho : R,\ \sigma : S,\ \rho' : R/S,\ \sigma' : S/R \implies \rho\,;\sigma' \sim \sigma\,;\rho'$

(ii) $\rho \sim \sigma \sim \tau \implies \rho \sim \tau$

(iii) $\rho \sim \sigma \implies \rho\,;\tau \sim \sigma\,;\tau$

(iv) $\rho \sim \sigma \implies \tau\,;\rho \sim \tau\,;\sigma$

Thus $\rho \sim \sigma$ means that ρ and σ differ by a (possibly empty) sequence of basic permutations. Therefore we know that the two coinitial reductions ρ and σ are

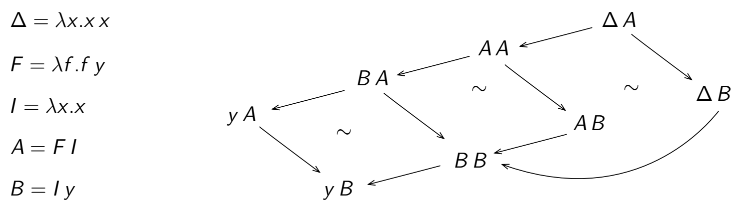

Fig. 6 Reductions equivalent by permutations from $\Delta\, A$ to $y\, B$

cofinal. But the converse may not be true. For instance on figure 5, the two reduction steps from $I(I\,x)$ to $I\,x$ are not equivalent by permutations. In [23, 7, 17, 31], a more computable definition of the equivalence by permutations is given. It involves the cube lemma and residuals of reductions which we skip here. There is also an alternative way of checking the equivalence by permutations with the use of the labeled λ-calculus.

Let U be a term of the labeled λ-calculus and $|U|$ be the same term where all labels are erased. Therefore $M = |U|$ is a term of the standard unlabeled λ-calculus. Similarly if ρ' is a reduction in the labeled λ-calculus, we write $|\rho'|$ for the same reduction in the standard λ-calculus. In a more sophisticated terminology, the $|\cdot|$ operator is a forgetful functor.

Theorem 4 *Let ρ and σ be two reductions starting from M. Let ρ' and σ' be two reductions starting from U in the pure labeled λ-calculus such that $M = |U|$, $\rho = |\rho'|$ and $\sigma = |\sigma'|$. Then*

$$\rho \sim \sigma \quad \Longleftrightarrow \quad \rho' : U \twoheadrightarrow V \ \wedge\ \sigma' : U \twoheadrightarrow V$$

Thus reductions equivalent by permutations corresponds to coinitial and cofinal reductions in the pure labeled calculus. For instance on figure 5, the difference between the unlabeled and labeled reduction graphs is subsumed by permutation equivalence. Similarly when $\Delta = \lambda x.\, xx$ and $\Omega = \Delta\Delta$, the two cofinal reductions $\Omega \to \Omega$ and $\Omega \to \Omega \to \Omega$ are not equivalent by permutations. It is also an easy exercise to check the permutation equivalences in figures 3, 4, 5, 6.

Definition 4 (prefix) Let ρ and σ be coinitial reductions. Then ρ is a prefix of σ up to permutations, written $\rho \le \sigma$, when there exists τ such that $\rho\,;\tau \ \sim\ \sigma$.

Proposition 4 *Let ρ and σ be two reductions starting from M. Let ρ' and σ' be two reductions starting from U in the pure labeled λ-calculus such that $M = |U|$, $\rho = |\rho'|$ and $\sigma = |\sigma'|$. Then*

$$\rho \le \sigma \quad \Longleftrightarrow \quad \rho' : U \twoheadrightarrow V \ \wedge\ \sigma' : U \twoheadrightarrow W \ \wedge\ V \twoheadrightarrow W$$

Thus the prefix ordering on unlabeled reductions corresponds to the lattice structure of reductions in the labeled λ-calculus. Therefore reductions of the unlabeled calculus have a push-out behaviour w.r.t. the category of coinitial reductions. Notice too that equivalence by permutations corresponds to antisymmetry of the prefix ordering. Furthermore, equivalence by permutations is left simplifiable, i.e. $\rho\,;\sigma \sim \rho\,;\tau$ implies $\sigma \sim \tau$.

Theorem 5 (standardization) *Let ρ be a reduction in the unlabeled λ-calculus. There exists a unique standard reduction ρ_{st} such that $\rho \sim \rho_{st}$.*

Therefore standard reductions whose reduction strategies are outside-in and left-to-right are canonical representatives of equivalence classes by permutations.

4 The history of redexes

When we explored the labeled λ-calculus, we mentioned that the name of redexes contains the names of their creators. In some sense, the history of a redex is contained in its name. If we start from a term with all subterms labeled with distinct atomic letters, the redexes with atomic names correspond to redexes already existing in the initial term. The redexes with composite names correspond to redexes created along reductions. Moreover we know by theorem 4 and proposition 1 that the names of redexes are preserved by permutations of reduction steps. The following proposition ensures that residuals of redexes are consistent with the permutation equivalence.

Proposition 5 *Let ρ and σ be two coinitial reductions starting at M and let R be a redex in M. Then $\rho \sim \sigma \implies R/\rho = R/\sigma$.*

Since the labeled calculus captures the history of reductions in the names of redexes, one cannot speak of a redex by itself in the unlabeled calculus. We have to add the reduction leading to the apparition of that redex. Therefore we write $\langle \rho, R \rangle$ when R is a redex in the final term of ρ. We call such a pair a h-redex, i.e. a redex and its history.

Definition 5 Let ρ and σ be two coinitial reductions, and let R and S be redexes in the final terms of ρ and σ. Then $\langle \sigma, S \rangle$ is a copy of $\langle \rho, R \rangle$, written $\langle \rho, R \rangle \le \langle \sigma, S \rangle$, and is defined as follows:

$$\langle \rho, R \rangle \le \langle \sigma, S \rangle \iff \exists \tau, \quad \rho\,;\tau \sim \sigma, \quad S \in R/\tau$$

Definition 6 Let ρ and σ be two coinitial reductions, and let R and S be redexes in the final terms of ρ and σ. Then $\langle \rho, R \rangle$ and $\langle \sigma, S \rangle$ are in a same family, written $\langle \rho, R \rangle \simeq \langle \sigma, S \rangle$, and is defined as follows:

$$\begin{array}{ll} \text{(i)} & \langle \rho, R \rangle \le \langle \sigma, S \rangle \implies \langle \rho, R \rangle \simeq \langle \sigma, S \rangle \\ \text{(ii)} & \langle \rho, R \rangle \le \langle \sigma, S \rangle \implies \langle \sigma, S \rangle \simeq \langle \rho, R \rangle \\ \text{(iii)} & \langle \rho, R \rangle \simeq \langle \sigma, S \rangle \simeq \langle \tau, T \rangle \implies \langle \rho, R \rangle \simeq \langle \tau, T \rangle \end{array}$$

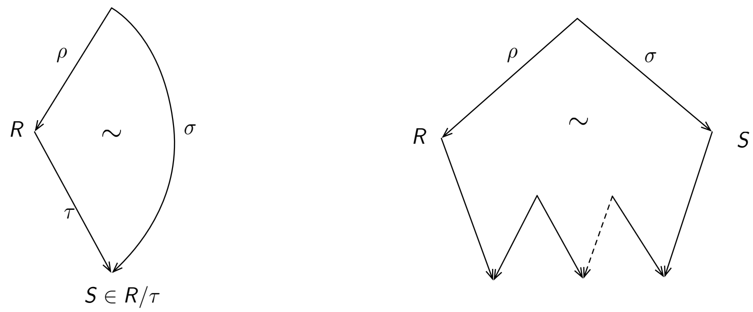

Fig. 7 Copies and the family relation on h-redexes

Clearly, when $\rho \sim \sigma$, the h-redex $\langle \sigma, S \rangle$ is a copy of h-redex $\langle \rho, R \rangle$. Thus the copy relation tracks redexes with their history up to permutation equivalence, and the family relation is the reflexive, symmetric and transitive closure of the copy relation on h-redexes. This is illustrated on figure 7 where a zigzag connects h-redexes $\langle \rho, R \rangle$ and $\langle \sigma, S \rangle$. On the example of figure 3, one can check that there is no alternative way of connecting the B redexes inside $(B\,A)$ and (A, B) with residuals along reductions. On that example, one can also check that the B redexes have all the same name γ. In fact, there are three families of h-redexes with names a, i and γ.

We now consider both labeled and unlabeled terms. We write $\mathsf{INIT}(U)$ when U is a labeled term with distinct letters on all subterms, and let write $\mathsf{name}(R)$ for the name of redex R.

Theorem 6 (redex history) *Let ρ and σ be two coinitial reductions, and let R and S be redexes in the final terms of ρ and σ. Let ρ' and σ' be two reductions starting from U in the pure labeled λ-calculus such that $M = |U|$, $\rho = |\rho'|$, $\sigma = |\sigma'|$, $R = |R'|$ and $S = |S'|$. Then*

$$\langle \rho, R \rangle \simeq \langle \sigma, S \rangle \quad \Longleftrightarrow \quad \mathsf{INIT}(U) \;\wedge\; \mathsf{name}(R') = \mathsf{name}(S')$$

Thus the families of redexes correspond to their names in the labeled setting. These families characterize all redexes, not only the ones appearing in the initial term. There are two extra properties that explain our intuition as described at the beginning of this article.

Theorem 7 (GFD - generalized finite developments) *If contraction of h-redexes is restricted to a finite number of families, the calculus strongly normalizes. Moreover all maximal developments are equivalent by permutations.*

The GFD theorem corresponds to strong normalization of the labeled calculus when only a finite number of redex names are permitted to be contracted [24]. The theorem may be seen as a more general version of the classic finite development theorem (see [13]) which insures finiteness and uniqueness of developments of redexes in the initial terms (i.e. redexes with zero-history).

Proposition 6 (canonical representatives) *Each family of h-redexes contains a unique canonical representative $\langle \rho_{st}, R \rangle$ such that ρ_{st} is a standard reduction of minimal length.*

The canonical representative is proved by considering an extraction process which we skip here [23]. Intuitively the name of a labeled redex contains the standard reduction giving the canonical representative. This theorem can also be proved by considering balanced paths and Girard's theory of geometry of interaction as in [3]. On the example of figure 3, one can check that the canonical representatives of families a, i, γ are $\langle \epsilon, \Delta A \rangle$, $\langle \epsilon, A \rangle$ and $\langle \rho_i, B \rangle$ where ϵ is the empty reduction and ρ_i is the one-step reduction contracting redex with name i. Canonical representatives for redexes may be seen at the origin of stability in the λ-calculus [26].

5 Other calculi

Combinatory logic is another simple calculus [16]. From the labeled λ-calculus, we may guess the following calculus for a labeled combinatory logic. Considering the abstract syntax trees of terms which are redexes, namely IM, KMN and $SMNP$, the only labels α, β, γ between internal nodes of $I\,x$, $K\,x\,y$ or $S\,x\,y\,z$ redexes are taken into account to form the new labels $[I:\alpha]$, $[K:\alpha,\beta]$, $\lceil S:\alpha,\beta,\gamma \rceil$ and $\lfloor S:\alpha,\beta,\gamma \rfloor$. Below, we abbreviate the notation for the substitution of variables x, y and z by writing $\{\mathbf{x} := \mathbf{x}^\alpha\}$ for $\{x := x^\alpha; y := y^\alpha; z := z^\alpha\}$.

$$M, N \ ::= \ x \ \mid \ S \ \mid \ K \ \mid \ MN \ \mid \ M^\alpha$$

$$I^\alpha x \ \longrightarrow \ x^{[I:\alpha]}$$

$$(K^\alpha x)^\beta y \ \longrightarrow \ x^{[K:\alpha,\beta]}$$

$$((S^\alpha x)^\beta y)^\gamma z \ \longrightarrow \ ((xz)(yz)\{\mathbf{x} := \mathbf{x}^{\lfloor S:\alpha,\beta,\gamma \rfloor}\})^{\lceil S:\alpha,\beta,\gamma \rceil}$$

$$M^\alpha\{x := P\} = M\{x := P\}^\alpha \qquad\qquad (M^\alpha)^\beta = M^{\alpha\beta}$$

Overlined and underlined labels of the labeled λ-calculus are replaced by boxed, underlined, overlined labels made of the kind of conversion and the labels on the spine of each redex. Notice that the internal node in the right hand side of the conversion rules are not labeled. This is purely conventional. The important labels are the ones which appear on the edges of the right-hand-sides, since these are the labels forming the new created redexes. Finally the I and K rules use boxed labels in order to avoid two contiguous overlined and underlined labels.

Take $\Delta = SII$. Then there is the following looping reduction $\Delta\Delta \twoheadrightarrow \Delta\Delta$ since:

$$SII(SII) \to I(SII)(I(SII)) \to SII(I(SII)) \to SII(SII) \to \cdots$$

But if Δ is labeled with $\underline{\Delta} = (S^a I^d)^b I^e$, $\Delta = (S^f I^i)^g I^j$, there is no longer a looping reduction:

$$
\begin{aligned}
\underline{\Delta}^c \Delta^h &\longrightarrow (I^{d\lfloor \alpha_0 \rfloor} \Delta^{\gamma_0} (I^{e\lfloor \alpha_0 \rfloor} \Delta^{\gamma_0}))^{\lceil \alpha_0 \rceil} && \alpha_0 = S{:}a,b,c \quad \gamma_0 = h\lfloor \alpha_0 \rfloor \\
&\twoheadrightarrow (\Delta^{\beta_1} \Delta^{\gamma_1})^{\lceil \alpha_0 \rceil} && \beta_1 = \gamma_0[I{:}d\lfloor \alpha_0 \rfloor] \ \gamma_1 = \gamma_0[I{:}e\lfloor \alpha_0 \rfloor] \\
&\twoheadrightarrow (I^{i\lfloor \alpha_1 \rfloor} \Delta^{\gamma_1} (I^{j\lfloor \alpha_1 \rfloor} \Delta^{\gamma_1}))^{\lceil \alpha_1 \rceil \lceil \alpha_0 \rceil} && \alpha_1 = S{:}f,g,\beta_1 \\
&\twoheadrightarrow (\Delta^{\beta_2} \Delta^{\gamma_2})^{\lceil \alpha_1 \rceil \lceil \alpha_0 \rceil} && \beta_2 = \gamma_1[I{:}i\lfloor \alpha_1 \rfloor] \ \ \gamma_2 = \gamma_1[I{:}j\lfloor \alpha_1 \rfloor] \\
&\twoheadrightarrow (I^{i\lfloor \alpha_2 \rfloor} \Delta^{\gamma_1} (I^{j\lfloor \alpha_2 \rfloor} \Delta^{\gamma_1}))^{\lceil \alpha_2 \rceil \lceil \alpha_1 \rceil \lceil \alpha_0 \rceil} && \alpha_2 = S{:}f,g,\beta_2 \\
&\twoheadrightarrow (\Delta^{\beta_3} \Delta^{\gamma_3})^{\lceil \delta_2 \rceil \lceil \alpha_1 \rceil \lceil \alpha_0 \rceil} && \beta_3 = \gamma_2[I{:}i\lfloor \alpha_2 \rfloor] \ \ \gamma_3 = \gamma_2[I{:}j\lfloor \alpha_2 \rfloor] \\
&\longrightarrow \cdots
\end{aligned}
$$

Similarly a labeled version of orthogonal term writing systems (OTRS) is considered in [28, 29]. In ORTS, labels on subterms of right hand sides of conversion rules are taken into account since there could be redexes in them. Suppose $L \longrightarrow R$ be a conversion rule (now L and R are for left hand sides and right hand sides). Therefore we use a diffusion operator $\alpha \cdot R$ with the name α of every left hand side L on the labels of the right hand side R.

$$
\boxed{
\begin{array}{ll}
M, N, L, R \ ::= \ x \ \mid \ f(M_1, M_2, \cdots M_n) \ \mid \ M^\alpha & \\
L \ \longrightarrow \ \lceil \alpha \rceil \cdot R\{\mathbf{x} := \mathbf{x}^{\lfloor \alpha \rfloor}\} & \alpha = \mathsf{name}(L) \\
M^\alpha\{x := P\} = M\{x := P\}^\alpha & (M^\alpha)^\beta = M^{\alpha\beta} \\
\alpha \cdot x = x \qquad \alpha \cdot f(M_1, M_2, \cdots M_n) = f^\alpha(\alpha \cdot M_1, \alpha \cdot M_2, \alpha \cdot M_n) &
\end{array}
}
$$

This labeled calculus corresponds to our labeled combinatory logic when the function symbols f of ORTS are I, K, S and application (where diffusion is restricted to the external labels of right hand sides). This calculus also correponds to the labels used for recursive program schemes in [32] which strongly inspired our work. Permutation equivalence of recursive program schemes is also presented in [7].

6 Conclusion

Permutation equivalence is a general property of any locally confluent calculus. A quite simple example is the calculus of derivations in context-free languages, where reductions are derivations and permutation equivalence corresponds to parse trees. A parse tree is indeed characterized by a unique left-to-right (standard) derivation. Coinitial/cofinal derivations with distinct parse trees are not equivalent and correspond to what is named ambiguity.

In the case of calculi with critical pairs (for instance term rewriting systems with linear left hand sides which may overlap), the notion of residuals is no longer defined when the contracted redexes R and S form a critical pair. But permutation equivalence can also be defined inside subsets of reductions without conflicting reductions (see [9]). This is also the case in the λ-calculus when η-conversion and δ-rules together with β-conversion are considered, but to our knowledge this has not been explored, except in Interaction Systems [5].

For process calculi, permutation equivalence, labeled calculi and h-redexes define causality inside reductions. Therefore there is a connection with Winskel's theory of event structures as proved in [10, 20] where labels are related to bisimulation in these process calculi. Labeled process calculi are also present in the theory of reversible calculus [12]. History-based flow information can also be related to the labeled calculus [8]. The incremental evaluations for makefiles in the Vesta system are also described with a labeled functional language [1].

Another motivation for the study of the labeled λ-calculus was to formalize the evaluation of λ-terms with optimal sharing[25]. These evaluations are easily defined on first-order rewriting systems (or combinatory logic). Shared reductions are then implemented with dags and a correspondence may be shown between labels and nodes of dags. But when there are bound variables and functions as in the λ-calculus, the situation is more complex as shown in the appendix of [33]. Lamping [19, 4] gave an algorithm for the evaluation in the λ-calculus with sharing. Kathail [18] gave another algorithm. In [14, 3] Lamping algorithm is connected to Girard's geometry of interaction. From Lamping algorithm, one can adapt the shared evaluation of λ-terms to a shared calculus of proofs nets in linear logic [15].

A final property of the λ-calculus is the existence of the sublattice of family complete reductions in which leftmost outermost reductions reach the normal forms (when they exist) in a minimal number of reduction steps. A striking property of family complete reductions is that the redexes contracted at each step are all residuals of a single redex, thus meaning that family complete reductions are complete with respect to copies of redexes. They exactly capture the maximum sharing possible when redexes are copies of a single redex. This leads to the optimality of Lamping algorithm when counting the number of reduction steps. This reduction is the analogous of the call-by-need evaluation in standard programming languages [27, 2].

References

1. Martín Abadi, Butler Lampson, and Jean-Jacques Lévy. Analysis and caching of dependencies. In *Proc. of the 1996 ACM SIGPLAN International Conference on Functional Programming*, pages 83–91. ACM Press, May 1996.
2. Zena M. Ariola, Matthias Felleisen, John Maraist, Martin Odersky, and Philip Wadler. A call-by-need lambda calculus. In *Proc. 22nd ACM Symposium on Principles of Programming Languages*, pages 2330–246, January 1995.
3. Andrea Asperti, Vincent Danos, Cosimo Laneve, and Laurent Régnier. Paths, computations and labels in the λ-calculus. In *Proceedings of the 10th Annual Symposium on Logic in Computer Science*, pages 426–610, 1994.
4. Andrea Asperti and Stefano Guerrini. *The Optimal Implementation of Functional Programming Languages*. Cambridge University Press, 1999.
5. Andrea Asperti and Cosimo Laneve. Interaction systems i: The theory of optimal reductions. *Mathematical Structures in Computer Science*, 4(4):457–504, September 1994.
6. Henk P. Barendregt. *The Lambda Calculus, Its Syntax and Semantics*. North-Holland, 1981.
7. Gérard Berry and Jean-Jacques Lévy. Minimal and optimal computations of recursive programs. In *Proc. of POPL'77. Journal of the ACM*, volume 26. ACM Press, 1979.
8. Tomasz Blanc. *Propriétés de sécurité dans le lambda-calcul*. PhD thesis, Ecole polytechnique, Palaiseau, November 2006.

9. Gérard Boudol. Computational semantics of term rewriting systems. In M. Nivat and J.C. Reynolds, editors, *Algebraic Methods in Semantics*, pages 169–236. Cambridge University Press, 1985.
10. Gérard Boudol and Ilaria Castellani. A non-interleaving semantics for ccs based on proved transitions. *Fundamentae Informaticae*, XI:433–453, 1988.
11. Alonzo Church. *The Calculi of Lambda-Conversion*. Princeton University Press, 1941.
12. Ioana Cristescu, Jean Krivine, and Daniele Varacca. A compositional semantics for the reversible pi-calculus. In *Proc. of the 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 388–397, 2013.
13. Haskell B. Curry and Robert Feys. *Combinatory Logic*. North-Holland, 1958.
14. Georges Gonthier, Martín Abadi, and Jean-Jacques Lévy. The geometry of optimal lambda reduction. In *Proc. of the 19th Conference on Principles of Programming Languages*, pages 15–26. ACM Press, January 1992.
15. Georges Gonthier, Martín Abadi, and Jean-Jacques Lévy. Linear logic without boxes. In *Proc. of the 7th IEEE Symposium on Logic in Computer Science*, pages 223–234. ACM Press, June 1992.
16. Roger Hindley. Combinatory reductions and lambda reductions compared. *Zeit. Math. Logik*, 23:169–180, 1977.
17. Gérard Huet and Jean-Jacques Lévy. Computations in orthogonal rewriting systems. In *Computational Logic – Essays in Honor of Alan Robinson*, pages 395–443. The MIT Press, Cambridge, MA, 1992.
18. Vinod Kathail. *Optimal interpreters for lambda-calculus based functional languages*. PhD thesis, MIT, EECS department, 1990.
19. John Lamping. An algorithm for optimal lambda-calculus reduction. In *Proceedings of the 17th Annual ACM Symposium on Principles of Programming Languages*, pages 16–30, 1990.
20. Cosimo Laneve. Distributive evaluations of lambda-calculus. *Fundam. Informaticae*, 20(4):333–352, 1994.
21. Jean-Jacques Lévy. *Réductions sures dans le lambda-calcul*. PhD thesis, Univ. of Paris 7, Paris, June 1974. (thèse de 3ème cycle).
22. Jean-Jacques Lévy. An algebraic interpretation of the lambda-beta-k-calculus; and an application of a labelled lambda-calculus. *Theoretical Computer Science 2 (1976), North Holland, pp.97-114.*, 2:97–114, 1976. also presented at the Rome Symposium on the λ-calculus, 1975.
23. Jean-Jacques Lévy. *Réductions correctes et optimales dans le lambda-calcul*. PhD thesis, Univ. of Paris 7, Paris, January 1978.
24. Jean-Jacques Lévy. Generalized finite developments. In Yves Bertot, G. Huet, J.-J. Lévy, and G. Plotkin, editors, *From Semantics to Computer Science: Essays in Honour of Gilles Kahn*. Academic Press, 1980.
25. Jean-Jacques Lévy. Optimal reductions in the lambda-calculus. In J.P. Seldin and J.R. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda-Calculus and Formalism*. Academic Press, 1980. On the occasion of his 80th birthday.
26. Jean-Jacques Lévy. Redexes are stable in the lambda-calculus. In H. Barendregt, Stefano Guerrini, and Adolfo Piperno, editors, *A special issue dedicated to Corrado Böhm for his 90th birthday, Computing with lambda terms*, volume 27 of *Mathematical Structures in Computer Science*, pages 738–750. 2009.
27. John Maraist, Martin Odersky, David N. Turner, and Philip Wadler. Call-by-name, call-by-value, call-by-need, and the linear lambda calculus. In *Electronic Notes in Theoretical Computer Science*, pages 41–62, March 1995.
28. Luc Maranget. Optimal derivations in orthogonal term rewriting systems and in weak lambda calculi. In *Proc. of the 18th Conference on Principles of Programming Languages*. ACM Press, 1991.
29. Luc Maranget. *La stratégie paresseuse*. PhD thesis, Univ. of Paris 7, Paris, 1992.
30. Dana S. Scott. Continuous lattices. In F.W. Lawvere, editor, *Toposes, algebraic geometry and logic*, volume 274 of *Lecture Notes in Mathematics*, pages 97–136. Springer, 1972.
31. Terese (M. Bezem, J. W. Klop, and R. de Vrijer eds). *Term Rewriting Systems*. Cambridge University Press, 2003.

32. Jean Vuillemin. *Proof Techniques for Recursive Programs*. PhD thesis, Stanford University, Computer Science Department, 1973.
33. Christopher P. Wadsworth. *Semantics and pragmatics of the lambda-calculus*. PhD thesis, Oxford University, 1971.