Paths, Computations and Labels in the λ-calculus*

Andrea Asperti
Dip. di Matematica, Bologna

Cosimo Laneve
INRIA Sophia-Antipolis

Abstract

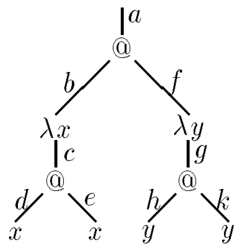
We provide a new characterization of Lévy's redex-families in the λ-calculus [12] as suitable paths in the initial term of the derivation. The idea is that redexes in a same family are created by "contraction" (via β-reduction) of a unique common path in the initial term. This fact gives new evidence about the "common nature" of redexes in a same family, and about the possibility of sharing their reduction. In general, paths seem to provide a very friendly and intuitive tool for reasoning about redex-families, as well in theory (using paths, we shall provide a remarkably simple proof of the equivalence between extraction [12] and labeling) as in practice (our characterization underlies all recent works on optimal graph reduction techniques for the λ-calculus [10, 7, 8, 1], providing an original and intuitive understanding of optimal implementations).

Finally, as an easy by-product of the path-characterization, we prove that neither overlining nor underlining are required in Lévy's labeling.

1 Introduction

Take the (labeled) λ-term $\Delta\Delta$, represented in the following picture:

*Partially supported by the ESPRIT Basic Research Project 6454 - CONFER. The work has been mostly carried out while the first author was at INRIA-Rocquencourt, and the second one was at the Dipartimento di Informatica of the University of Pisa.



Consider the application node labeled by c. Our question is: "is it possible that this application node will be ever involved in a β-reduction?". In order to solve the question we must look for a λ-node to match against the application. We eventually start our search towards the left son of the @-node (this is the *principal port* of the application, in Lafont's terminology [9, 2], that is the only port where we may have interaction with a dual operator). Thus, we pass the edge labeled by d and we find a variable. If the variable was free we had finished: no redex involving our application could possibly exist. Since the variable is bound, the "control" is passed back to its binder, that is the λ-node labeled by b in the picture. Indeed, a β-reduction involving the binder will replace the variable with the argument of the application, and we must continue our search inside this argument. Hence we must pose a symmetrical question about this λ-node, namely: "is it possible that this λ-node will be ever involved in a β-reduction?". Since the question concerns a λ-node, we must now start traveling towards the root (note that this is still the principal port of the λ-node, according to Lafont). This time, we find an application. Moreover we enter the application at its principal port, thus we have found a redex. The last question is solved positively, and we must resume the previous one, looking into the argument of the application. Thus, we pass f, and we find a λ. This λ-node is reached at its principal port, so we have finally found a (virtual) redex for the original @-node.

During this process, we have described a "path" dbf in t. If we suppose that bound variables are explicitly connected to (positive, auxiliary ports of) their respective binders (an idea going back to Bourbaki), the "path" is indeed connected. In the following, we shall always make this hypothesis, even if we shall not explicitly draw the connections in the pictures. Note also that, by firing the redex b, we "contract" the path dbf into a single redex-edge. This is the "actual" redex, corresponding to its virtual path-description dbf.

Let us consider another example, starting with the @-node labeled by g, in the picture above. We travel along h, then back to the binder for y, and up along f. We have found a @-node, but we entered this node at its negative, auxiliary port (not its principal port!). So we did not find a redex, and we must open a new session, looking for a redex involving this @-node. This is immediately found: it is the redex b. Hence we resume the previous search. Since the "control" was coming from the argument of the @-node labeled by a, we must pass it to some of the variables bound by the λ-node in the redex b, that is d or e (note the non-determinism of the search algorithm, at this point). Suppose to follow the first possibility. We have found the same redex dbf of the previous computation (traveling in the opposite direction), and we may resume the first question. We must go down to e, up to the binder for x, up again along the redex b and finally down along f. We have described the path $hfbdebf$ beginning at the principal port of a @-node and ending to the principal port of a λ-node. The reader may convince himself that this path corresponds to the unique redex created after two reductions of $\Delta\Delta$ (see also example 5.2). Indeed, by firing first the redex b and then the (newly created) redex dbf, we contract the path $hfbdebf$ into a single redex-edge.

Actually, the search algorithm is more contrived of what we have described. In particular, the non deterministic choice at the level of bound variables (when traveling down through a λ) is not always free.

Our aim, for the moment, was just to provide the main intuitions behind the path-description of "virtual" redexes in a λ-term t (i.e. redexes which can be created along some reduction from t). Every "virtual" redex t defines a suitable "legal" path in t from an application node to a λ-node.

Do different "virtual" redexes define different paths? The answer is no, due to a duplication problem. Consider a term $t = (\lambda x.M)N$, and suppose to have a redex r inside N. This redex may have several residuals in $M[N/x]$, and all these different residuals define the same path in t. This fact suggests that we can reasonably hope to have an injection from "virtual" redexes to paths, up to some notion of "sharing".

This is actually the case. Formally, we have a *bijective correspondence* between families of redexes in the sense of Lévy [12] and (a suitable class of *legal*) paths. In other words,

> *two redexes are in a same Lèvy's family, if and only if their associated paths in the initial term of the derivation coincide.*

The first hints behind the notion of legal path were already in Lamping's work [10]. In particular, every *prerequisite chain* is always a prefix or a suffix of our legal paths. However, in order to characterize every "virtual" redex as a legal path, we must suitably compose prerequisite chains, that is the not trivial part of the work (and one of our original contributions). Moreover, Lamping did never establish a clear correspondence between prerequisite chains and Lévy's labeling.

An (implicit) account of this correspondence is already in [7]. Unfortunately the paper is rather cryptic (you will have to deal with the *bus notation*). Moreover, the notion of legal path is based on Lamping's control operators, and it has pretty operational flavor that makes it difficult to understand.

Up to our knowledge, the first ones to pursue the program of describing "virtual" redexes (computations) by means of paths have been Danos and Regnier [14, 5]. However, they did not remark the relation between paths and redex families (optimal reductions). The correspondence between Danos and Regnier's *regular* paths and our legal paths has been recently explained in [4].

The structure of the paper is the following. We shall start with introducing the notion of redex-family in the λ-calculus [12], devoting a particular attention to the so called *extraction process* (section 3) and to Lévy's labeling (section 4). In Section 5 we shall define the notion of *path associated with a redex*, by relying on its label; we shall also prove that neither overlining non underlining are required in Lévy's labeling. Section 6 is devoted to a first application of the path characterization of the family relation: we shall provide a remarkably simple proof of the relation between labeling and extraction.

Obviously, not every path in the initial term of a derivation is associated with a redex-family. The final part of the paper provides a complete characterization of these paths (*legal* paths), independent from labels.

2 The family relation

In 1978, Lévy introduced the notion of redex family in the λ-calculus, with the aim to formally capture an *intuitive* idea of *optimal sharing* between "copies" of a same redex. For a long time, no λ-calculus implementation has been able to achieve the theoretical performance fixed by Lévy (see [6]), and it is only in recent years [10, 7] that this problem has been finally solved (see also [3, 11] for a generalization of these results to a wider class of higher order term rewriting systems).

In order to comfort his notion of family, Lévy proposed several alternative definitions, inspired by different perspectives, and proved their equivalence.

The most abstract approach to the notion of family [12] is the so called *zig-zag*. In this case, duplication of redexes is formalized as residuals modulo permutations. In particular, a redex u with history σ (notation σu) is a *copy* of a redex v with history ρ iff $\rho v \leq \sigma u$ (i.e., there exists τ such that $\sigma = \rho\tau$ up to permutation equivalence, and u is a residual of v after τ). The family relation $\simeq$ is then the symmetric and transitive closure of the copy-relation (pictorially, this gives rise to the "zig-zag"). Note that, intuitively, the family relation has to be an equivalence relation.

Another approach is that of considering the *causal history* of the redex. Intuitively, two redexes can be "shared" if and only if they have been created "in the same way" (or, better, their *causes* are the same). This is formalized by defining an *extraction relation* over redexes (with history) σu, which throws away all the redexes in σ that have not been relevant for the creation of u. The canonical form we obtain at the end of this process essentially expresses the causal dependencies of u along the derivation (we may deal with causal *chains* instead of *partial orders* since *only standard derivations* are considered). The extraction relation will be formalized in the next section.

The most "operational" approach to the family relation is based on a suitable labeled variant of the λ-calculus [12], described in the next section. The idea of labels is essentially that of marking the "points of contact" created by reductions. In particular, labels grow along the reduction, keeping a trace of its history. Two redexes are in a same family if and only if their labels are identical.

The equivalence between zig-zag and extraction is not particularly problematic [13]. On the contrary, the proof of the equivalence between extraction (or zig-zag) and labeling is much more difficult, and it forms the core of Lévy's Thesis [12]. The relation between labels and "paths" described in this paper allow us to provide a much simpler proof of this fact (see section 6). Actually, in our experience, the approach to the family relation based on "paths" provides the most friendly and intuitive tool for reasoning about redex-families, as well in theory as in practice (optimal implementation).

3 Extraction

The idea behind the extraction relation, is that of associating to every redex u created along some derivation σ, the *shortest* (standard) derivation which is needed to the creation of (a "copy" of) u. The interesting result is that this "canonical" derivation is unique, and can be effectively computed via a confluent and strongly normalizing process.

A few preliminary definitions are needed in order to address formally the extraction relation. We shall denote *β-redexes* $@(\lambda x.\, t_1,\, t_2)$ through the *access path* to the corresponding subexpression, defined as follows:

- ε (the empty string) is the access path of the root;
- if u is the access path of $\lambda x.\, t$ or $@(t,\, t')$ then $u \cdot 1$ is the access path of t and $u \cdot 2$ is the access path of t'.

Reductions will be represented by the access paths of the corresponding redex in bold or by $t \xrightarrow{u} t'$, when we want to emphasize the initial and final expressions. Sequences of reductions, called *derivations*, will be ranged over by $\sigma, \rho, \cdots$. Again the notation $t \xrightarrow{\sigma}\!\!\!\!\rightarrow t'$ will emphasize initial and final expressions.

We say that two derivations are *disjoint* if they contract redexes into disjoint subexpressions. Let $t \xrightarrow{u} t'$, $@(\lambda x.\, t_1,\, t_2)$ be the redex in t at occurrence u and $\rho = \mathbf{u}_1; \cdots; \mathbf{u}_n$ be a derivation starting at t'. Then we say that ρ is *internal to the functional argument* of $\mathbf{u}$ if for every $1 \leq j \leq n$:

$$\exists u'_j.\ u_j = u \cdot 1 \cdot u'_j \qquad \textbf{and} \qquad t_1 \xrightarrow{u'_1; \cdots; u'_j}\!\!\!\!\rightarrow$$

(that is ρ reduces redexes inside t_1). Under the same hypothesis we say that ρ is *internal to the i-th instance of the argument* of $\mathbf{u}$ if

$$\forall j.\ u \cdot w \preceq u_j$$

where w is the access path in t_1 of the i-th occurrence from the left of the variable x. Thus let $t \xrightarrow{u} t' \xrightarrow{\rho}\!\!\!\!\rightarrow$ and assume that ρ is internal to the i-th instance of the argument of $\mathbf{u}$. Then the reduction $\rho \| u$ (ρ *parallelized* by u), is inductively defined by

$$\begin{array}{rcll} \underline{\emptyset} \,\|\, u & = & \underline{\emptyset} & \\ (\mathbf{v}; \rho) \,\|\, u & = & (\mathbf{v}'/\mathbf{u}); (\rho/\mathbf{W}) \| (u/\mathbf{v}') & \qquad v \in v'/\mathbf{u}\,, \quad W = v'/(\mathbf{u}; \mathbf{v}) \end{array}$$

where $\underline{\emptyset}$ is the empty derivation and $v/\mathbf{u}$ denotes the *residuals* of the redex v after the contraction $\mathbf{u}$. A way to formalize residuals of a redex v w.r.t. $t \xrightarrow{u} t'$ is by marking v and taking all the marked redexes in t'. Notice that $v/\mathbf{u}$ is in general a (possibly empty) set of redexes. An alternative definition can be found in [13]. Remark that the derivation $\rho\|u$ is actually *parallel*, in general. That is each step is a contraction of a *set* of redexes. The well-definition of $\rho\|u$ follows by the Parallel Move Lemma in [13].

Let $\sigma/\mathbf{u}$ (the residual of a derivation σ w.r.t. $\mathbf{u}$) be inductively defined by:

$$\sigma/\rho \;=\; \begin{cases} \underline{\emptyset} & \textit{if } \sigma = \underline{\emptyset} \\ (\sigma'/\rho); v/(\rho/\sigma') & \textit{if } \sigma = \sigma'; \mathbf{v}. \end{cases}$$

Then we denote with $\mathbf{u} \sqcup \sigma$ the derivation $\mathbf{u}; \sigma/\mathbf{u}$.

Definition 3.1 (**extraction relation**) The extraction relation is the union of the following four relations:

1. $\rho; \mathbf{u}; \mathbf{v} \;\; \rhd_1 \; \rho; \mathbf{v}'$, if $v \in v'/\mathbf{u}$;
2. $\rho; (\mathbf{u} \sqcup \sigma) \;\; \rhd_2 \; \rho; \sigma$, if $|\sigma| \geq 1$ and $\mathbf{u}$ and σ are disjoint reductions;
3. $\rho; (\mathbf{u} \sqcup \sigma) \;\; \rhd_3 \; \rho; \sigma$, if $|\sigma| \geq 1$ and σ is internal to the function part of $\mathbf{u}$;
4. $\rho; \mathbf{u}; \sigma \;\; \rhd_4^i \; \rho; \sigma'$, if $|\sigma| \geq 1$ and σ is internal to the i-th instance of the argument of $\mathbf{u}$ and $\sigma'/\mathbf{u} = \sigma\|u$.

We will denote with $\trianglerighteq$ the transitive and reflexive closure of $\rhd$. ∎

Theorem 3.2 [13] The relation $\trianglerighteq$ is confluent and strongly normalizing. ∎

Definition 3.3 Every derivation σu which is standard and in normal form w.r.t. $\trianglerighteq$ will be called *canonical.* ∎

The intuitive idea, is that two redexes are in a same family if and only if their canonical derivations coincide. In Section 6 we shall provide an original proof of the equivalence between extraction and labeling, by means of paths.

4 Labeling

The *labeled λ-calculus* is an extension of the λ-calculus proposed by Lévy in [12]. In order to avoid some annoying problems concerning the associativity of the concatenation operator over labels, our presentation will be slightly different (but equivalent) w.r.t. Lévy's one. In particular, we shall delay the concatenation of labels until it is required for firing a redex.

Definition 4.1 Let $L = \{a,\, b,\, \cdots\}$ be a denumerable set of *atomic labels*. The set $\mathbf{L}$ of *labels*, ranged over by $\ell,\, \ell_1, \cdots$, is defined by the following rules:

$$L \quad | \quad \ell_1\ell_2 \quad | \quad \underline{\ell} \quad | \quad \overline{\ell}$$

The operation of concatenation $\ell_1\ell_2$ will be assumed to be associative. The set $\mathbf{L}_p$ of *proper labels* contains exactly all those labels $\ell \in \mathbf{L}$ such that ℓ is atomic or $\ell = \underline{\ell'}$ or $\ell = \overline{\ell'}$. $\mathbf{L}_p$ will be ranged over by $\alpha,\, \beta, \cdots$.
Given a label ℓ, $\mathtt{at}(\ell)$ is the set of all its *atomic* (sub)labels. ■

For instance, $\ell = \underline{a\overline{bab}c}$ is a label, and $\mathtt{at}(\ell) = \{a, b, c\}$.

Let us come to the formal definition. Labeled λ-terms are those obtained by the following syntax:

$$t \;::=\quad x \quad | \quad \lambda x.\, t \quad | \quad @(t,\, t') \quad | \quad \alpha(t)$$

where $\alpha \in \mathbf{L}_p$. The β-reduction is now a set of rules: for every n-tuple $\alpha_1, \cdots, \alpha_n$, we have the following rule:

$$@(\alpha_1(\cdots(\alpha_n(\lambda x.\, X)\cdots), Y) \quad\longrightarrow\quad \overline{\alpha_1\cdots\alpha_n}(X[{}^{\underline{\alpha_1\cdots\alpha_n}(Y)}/_{x}])$$

When a redex is fired, a label ℓ is captured between @ and λ: this is the *degree* of the redex. Then, every possible interaction created in the rhs of the rewriting rule must be suitably "marked" with ℓ, in order to keep a trace of the history of the creation. In the case of λ-calculus, there are two ways to create redexes: "towards the top", i.e. a redex in which the outermost symbol of the functional part of the rule is involved, and "towards the bottom", i.e. a redex in which the outermost symbol of the argument part is involved. Overlinings and underlining essentially express this double possibility.

We will say that a given subexpression t' occurring in t has label $\alpha_1\cdots\alpha_n$ if $t = \alpha_1(\cdots\alpha_n(t')\cdots)$ or $\alpha_1(\cdots\alpha_n(t')\cdots)$ is the body of an abstraction or an argument of an application.

Definition 4.2 (**Initial labeling**) Let t be a labeled λ-expression. The predicate $\mathtt{INIT}(t)$ is true if and only if the labels of all subterms of t are atomic and pairwise different. ■

Notation 4.3 Let t be a λ-expression: we shall denote with t_I the labeled λ-expression such that $\mathtt{INIT}(t_I)$ and equal to t up to the labeling. Let $t \xrightarrow{\sigma}\!\!\!\!\rightarrow t'$ and u be a redex in t'. We shall denote with $\partial_t^\sigma(u)$ the degree (i.e. the label) of u along the labeled reduction isomorphic to σ and starting at t_I. ■

5 Labels as paths

Labels provide a very simple approach to the notion of computation as a travel along a path. In particular, every label trivially define a path in the initial term of the computation. The interesting problem will be to provide an independent characterization of these paths (see Section 7).

Let us assume here the graph representation of terms mentioned in the introduction (i.e. bound variables are supposed connected to the respective binders). This assumption will be crucial for our definition of paths.

Consider an expression t such that $\mathtt{INIT}(t)$. Every edge in t is labeled with a different atomic symbol, so we may call each edge by its label.

Definition 5.1 If ℓ is a label of an edge generated along some reduction from t, the *path* of ℓ in t is inductively defined as follows:

$$\begin{array}{rcl} \mathtt{path}(a) & = & a \\ \mathtt{path}(\ell_1\ell_2) & = & \mathtt{path}(\ell_1)\cdot\mathtt{path}(\ell_2) \\ \mathtt{path}(\overline{\ell}) & = & \mathtt{path}(\ell) \\ \mathtt{path}(\underline{\ell}) & = & (\mathtt{path}(\ell))^r \end{array}$$

where "$\cdot$" means concatenation (it will be omitted in the following) and $(\varphi)^r$ is as φ but reversed. ■

It is easy to prove by induction on the length of the derivation generating the label ℓ that the previous definition is sound, i.e. that $\mathtt{path}(\ell)$ is indeed a path in t (it simply follows by the labeled β-contraction).

Example 5.2 Consider again the labeled term $\Delta\Delta$ of the introduction. After two labeled β-reduction, you obtain a redex with degree $\ell = h\underline{d\underline{b}f}e\underline{b}f$. Thus $\mathtt{path}(\ell) = hfbdebf$, that is the path we obtained in the introduction for the application labeled with g. ■

The interesting fact is that different degrees define different paths in the initial term. A preliminary proposition is required, concerning the structure of labels.

Proposition 5.3 Let $\alpha_1 \cdots \alpha_n$ be a label generated along some derivation σ starting at t (every α_i is a *proper* label). Then

1. n is odd; for every i odd, α_i is atomic; for every i even, $\alpha_i = \overline{\ell}$ or $\alpha_i = \underline{\ell}$;

2. if $\alpha_{2i} = \overline{\ell}$ then α_{2i-1} marks the output edge of the node @ (in t) determinated by the leftmost label in ℓ. α_{2i+1} marks the input port of the abstraction (in t) individuated by the rightmost label in ℓ.

3. if $\alpha_{2i} = \underline{\ell}$ then α_{2i-1} labels an edge in t incoming into the bound port of the abstraction individuated by the rightmost atomic label in ℓ. α_{2i+1} marks the edge of the second argument of the application determinated by the leftmost label in ℓ.

Proof: Easy induction on the length of the derivation. ■

Remark 5.4 Observe that, if ℓ is a label relative to a redex, it may appear inside another label only if it is overlined or underlined, and surrounded by atomic labels. ■

An easy consequence of the above proposition is that both underlinings and overlinings can be safely omitted in Lévy's labeling (a property that was already known to Gonthier and Lévy, but never published).

Corollary 5.5 Neither overlining nor underlining are needed in Lévy's labeling.

Proof: (sketch) Remove all underlinings and overlinings from a label ℓ. This can be formalized by a function `flat` that looks like `path` except for $\texttt{flat}(\underline{\ell}) = \texttt{flat}(\ell)$. The initial structure can be uniquely retrieved by working "inside out", using Proposition 5.3. That is, starting from atomic labels relative to redexes, we overline or underline them according to the surrounding labels (since we know the initial labeling of the term, the previous step is deterministic). Then repeat the process with the structured labels yielded so far. ■

Proposition 5.6 The function `path` is injective over labels generated along derivations starting from an expression owning `INIT`.

Proof: Let ℓ_1 and ℓ_2 be two different labels. The proof is by induction on the structure of ℓ_1. The case ℓ_1 atomic is easy. Thus, assume that $\ell_1 = \alpha_1 \dots \alpha_n$. If $\ell_1 = \ell_2 \ell$ (or vice versa) the thesis follows trivially, since the two paths have different lengths. Note that this case is not possible when one of the two labels ℓ_1 or ℓ_2 is relative to a redex, by Remark 5.4. Suppose then $\ell_2 = \beta_1 \dots \beta_m$, and let k be the first index such that $\alpha_k \neq \beta_k$. Three subcases are possible.

- α_k is atomic. By Proposition 5.3, also β_k must be atomic. Since they are different, the two paths diverge here.
- $\alpha_k = \overline{\ell'_1}$. Then $\beta_k = \overline{\ell'_2}$ by Proposition 5.3 (note that β_k cannot be underlined, since otherwise, $\alpha_{k-1} \neq \beta_{k-1}$). Then we use the inductive hypothesis over ℓ'_1 and ℓ'_2 (note in particular that the two paths must really diverge and they cannot be one an initial subpath of the other).
- $\alpha_k = \underline{\ell'_1}$. Analogous to the previous one. ■

6 The equivalence between extraction and labeling

In this section we show a first application of the previous results, by providing a simple and original proof of the equivalence between extraction and labeling. This correspondence is formally stated by the following theorem.

Theorem 6.1 Let $t \stackrel{\rho}{\twoheadrightarrow} t^\rho$ and $t \stackrel{\sigma}{\twoheadrightarrow} t^\sigma$. Then $\rho u \simeq \sigma v$ if and only if $\partial_t^\rho(u) = \partial_t^\sigma(v)$. ■

The easy part of the above equivalence is that redexes in the same family (w.r.t. the extraction relation) have the same degree. Indeed it is enough to show that the standardization and the extraction processes do not change labels of redexes in the ending term of a derivation (see [12]).

The converse is much more contrived. Our proof relies on the two lemmas below.

Lemma 6.2 (**The First Redex Lemma**) Let σv be a canonical derivation. Let u be the leftmost outermost redex in t such that $\partial_t^\emptyset(u) \in \mathtt{at}(\partial_t^\sigma(v))$. Then u is the first redex fired by $\sigma; \mathbf{v}$. ■

Lemma 6.3 (The Contracted Label Lemma) Let $t \xrightarrow{w} t^+ \xrightarrow{\rho}\!\!\!\!\rightarrow t' \xrightarrow{u}$ and $t \xrightarrow{w} t^+ \xrightarrow{\sigma}\!\!\!\!\rightarrow t'' \xrightarrow{v}$ be two canonical derivations. Then

$$\partial_t^{w;\rho}(u) \;=\; \partial_t^{w;\sigma}(v) \;\Rightarrow\; \partial_{t^+}^{\rho}(u) \;=\; \partial_{t^+}^{\sigma}(v) \qquad \blacksquare$$

The formal proof of the two lemmas above will be given in the following subsections. However, the intuitive idea is very simple. In the case of the first lemma, every (atomic) redex traversed by the path associated with the degree of v is needed for its creation. So the leftmost outermost will be the first one fired along the canonical derivation. In the case of the second lemma, we already know that if $\partial_{t^+}^{\rho}(u) \;\neq\; \partial_{t^+}^{\sigma}(v)$ they define different paths in t^+. Now, the only possibility that these paths get a same labeling when starting with t_I, is that they are isomorphic paths inside different instancies of the argument of the redex w, which have been duplicated along its reduction. But this case is excluded by canonicity.

Let us come to the proof of the main theorem.

Proof of Theorem 6.1: As we have remarked, the difficult direction to prove is

$$\partial_t^{\rho}(u) \;=\; \partial_t^{\sigma}(v) \;\Rightarrow\; \rho u \;\simeq\; \sigma v$$

So, assuming the other direction, we can safely reduce to consider the case when ρu and σv are canonical. As a matter of facts, the above implication reduces to

$$\rho u \;\neq\; \sigma v \;\Rightarrow\; \partial_t^{\rho}(u) \;\neq\; \partial_t^{\sigma}(v)$$

(see [12]). Assume that ρ is shorter than σ. By induction on ρ:

(**case** $\rho = \underline{\emptyset}$) Here u has to be a redex in the initial term t (thus its label is atomic). There are two subcases: $\sigma = \underline{\emptyset}$ and $\sigma = \mathbf{v}'; \sigma'$. The first subcase is immediate because $u \neq v$ implies that their degree is different. In the second case, $\partial_t^{\sigma}(v)$ cannot be atomic, since otherwise it would individuate a redex in t, and hence σv would not be in normal form w.r.t. $\underline{\triangleright}$.

(**case** $\rho = \mathbf{w}; \rho'$) Let $\sigma = \mathbf{w}'; \sigma'$. There are two subcases: either $w \neq w'$ or $w = w'$. In the first case, one between w and w' will be external to the other or they are disjoint. In both cases, by Lemma 6.2, the leftmost outermost redex in $\mathtt{at}(\partial_t^{\rho}(u))$ and $\mathtt{at}(\partial_t^{\sigma}(v))$ are different, and thus $\partial_t^{\rho}(u) \neq \partial_t^{\sigma}(v)$.

If $w = w'$, let $t \xrightarrow{w} t^+$. Since $\rho u \neq \sigma v$, we must have $\rho' u \neq \sigma' v$. By inductive hypothesis, $\partial_{t^+}^{\rho}(u) \neq \partial_{t^+}^{\sigma}(v)$ and, by Lemma 6.3, $\partial_t^{w;\rho}(u) \neq \partial_t^{w;\sigma}(v)$. $\blacksquare$

The original proof given by Lévy ([12], pp. 68-113) is *much* more contrived. It is based on a complex notion of labeled subcontext, and it takes over 40 pages in his thesis. It is interesting to note that his induction on the length of the derivation works in the opposite direction, i.e. considering the tail redex of the canonical derivation. The main reason behind our simpler proof is that we take advantage of some structural properties of labels that Levy did not notice at that time (in particular, the fact that they are *connected* along a path).

6.1 Proof of The First Redex Lemma

Proof of Lemma 6.2: By induction on the length of the derivation σ. The basic case is obvious. Let σ be $t \xrightarrow{u} t' \xrightarrow{\sigma'}\!\!\!\!\rightarrow$, $a = \partial^{\emptyset}_{t}(u)$, w be the first redex in $\sigma' v$ and $t_I \xrightarrow{u_I} t'_\star$ be the labeled reduction isomorphic to $t \xrightarrow{u} t'$. By induction, $\partial^{\emptyset}_{t'}(w) \in \mathtt{at}(\partial^{\sigma'}_{t'}(v))$ and notice that $\partial^{\sigma}_{t}(v) = \partial^{\sigma'}_{t'}(v)[\mathtt{s}]$, where $\mathtt{s}$ is the substitution of labels in t'_I with those marking the corresponding edge in $t'_\star$. We distinguish two cases.

(A) The (application relative to the) redex w is external or to the left of u. Since σv is canonical, w has been eventually created by u. Namely, we are in a situation where the subexpression at w is

$$@(@(\lambda x.\, t_1,\, t_2),\, t_3)$$

and the contraction $u = @(\lambda x.\, t_1,\, t_2)$ produces an expression $\lambda y.\, t_4$. Observe that $a \in \mathtt{at}(\partial^{\emptyset}_{t'}(w)[\mathtt{s}])$. Now, since $\partial^{\emptyset}_{t'}(w) \in \mathtt{at}(\partial^{\sigma'}_{t'}(v))$ and $\partial^{\sigma}_{t}(v) = (\partial^{\sigma}_{t}(v))[\mathtt{s}]$, it results that $a \in \partial^{\sigma}_{t}(v)$.

(B) The other case is when the reduction $\mathbf{w}$ is internal to u (otherwise it is easily proved that σv could not be canonical). Suppose $u = @(\lambda x.\, t_1,\, t_2)$. The derivation $\sigma' v$ cannot be internal to t_1 or t_2, since otherwise u could be removed from σ. There are two subcases: (**B.1**) $\sigma' v$ is not internal to the subexpression at u in the rhs of $\mathbf{u}$ or (**B.2**) it is internal. **B.1** is similar to **A** and thus omitted. **B.2** has two further cases: w is internal to one of the instances of t_2 or w is internal to t_1 (the case when w is such that the application @ involved in the contraction $\mathbf{w}$ comes from t_1 while the abstraction λ comes from t_2 between t_1 and t_2 is obvious).

When w is internal to one of the instances of t_2 then $\sigma' v = \sigma_1; \mathbf{w}'; \sigma_2$ where σ_1 is internal to t_2 and w' is the access path of the instance of

t_2. This because $\sigma' v$ must also contract redexes in t_1 and, in order that this is possible, due to the standardization, it can only contract redexes external to w that are created by σ_1.

When w is internal to t_1, since $\sigma' v$ must eventually fire redexes involving (some instance of) t_2, there are two ways. $\sigma' v = \sigma_1; \mathbf{w}'; \sigma_2$, with σ_1 internal to t_1 and $\mathbf{w}'$ is the access path of one of the instances of t_2; or $\sigma' v = \sigma_1; \mathbf{w}^+; \sigma_2$, with σ_1 internal to t_1 and $\mathbf{w}^+$ is internal to one of the instances of t_2. In the latter alternative we can reduce to the previous subcase by the constraint of canonicity. Namely $\sigma_2 = \sigma_2'; \mathbf{w}'; \sigma_2''$ with $\mathbf{w}'$ "on the border" of the instance of t_2.

Thus, in every case, $\sigma'; \mathbf{v} = t \xrightarrow{\sigma_1} t^+ \xrightarrow{w';\sigma_2}$ where $|\sigma_1| \geq 1$ and $\mathbf{w}'$ is "on the border" of an instance of t_2. We recall that, according to the labeled β-reduction, the label of w' must contain $\underline{a}$ as sublabel.

Let $\sigma_2 = \sigma_2' v$. Then, by inductive hypothesis, $\partial^{\emptyset}_{t^+}(w') \in \mathtt{at}(\partial^{w';\sigma_2'}_{t^+}(v))$ and, according to the assumptions over w', $a \in \mathtt{at}(\partial^{\sigma_1}_{t^+}(w'))$.

This immediately yields the statement of the lemma. ∎

6.2 Proof of The Contracted Label Lemma

Proof of Lemma 6.3: Take the two canonical derivations $(\mathbf{w}; \rho)u$ and $(\mathbf{w}; \sigma)v$ and $t \xrightarrow{w} t^+$. The hypothesis is $\partial^{\rho}_{t^+}(u) \neq \partial^{\sigma}_{t^+}(v)$, that is $\partial^{\rho}_{t^+}(u)$ and $\partial^{\sigma}_{t^+}(v)$ define two different paths φ and ψ in t^+ (Proposition 5.6).

Let $t_I \xrightarrow{w_I} t^+_\star$ be the labeled reduction isomorphic to $t \xrightarrow{w} t^+$. Then $\partial^{w;\rho}_t(u) = (\partial^{\rho}_{t^+}(u))[\mathtt{s}]$ and $\partial^{w;\sigma}_t(v) = (\partial^{\sigma}_{t^+}(v))[\mathtt{s}]$, where $\mathtt{s}$ is the substitution replacing every atomic label in t^+_I with the label of the corresponding edge in $t^+_\star$.

By contradiction assume that $\partial^{w;\rho}_t(u) = \partial^{w;\sigma}_t(v)$. Take the beginning of $\partial^{\rho}_{t^+}(u)$ and $\partial^{\sigma}_{t^+}(v)$ and start to compute $\varphi_t = \mathtt{path}((\partial^{\rho}_{t^+}(u))[\mathtt{s}])$ and $\psi_t = \mathtt{path}((\partial^{\sigma}_{t^+}(v))[\mathtt{s}])$. Let $@(\lambda x.\, t_1,\, t_2)$ be the redex at w in t. There are several cases:

1. φ_t and ψ_t begin in the context of the redex at w or in t_1. If φ_t never enters into t_2 then also ψ_t must never enter because the substitution $\mathtt{s}$ is bijective on the labels of the edges that are outside the residuals of t_2 uniquely identify the edges. This implies $\varphi = \psi$ and, by Proposition 5.6, $\partial^{\rho}_{t^+}(u) = \partial^{\sigma}_{t^+}(v)$, which contradicts the hypothesis.

2. φ_t and ψ_t begin in the context of the redex at w or in t_1 and eventually enter into t_2. There are two subcases:

- φ_t and ψ_t enter from the application of the redex at w. Since $\partial_{t+}^{w;\rho}(u) = \partial_{t+}^{w;\sigma}(v)$, φ_t and ψ_t must enter from the same bound edge of the abstraction $\lambda x.\, t_1$ because $\mathbf{s}$ is bijective on such edges. This means that φ and ψ enter in the same copy of t_2 in t^+. The contradiction follows by reiterating the reasoning.
- φ_t and ψ_t enter from a bound edge of an abstraction outer than w. This case is not possible, since it implies that the outer abstraction is fired along $(\mathbf{w};\rho)u$ or $(\mathbf{w}\sigma)v$, contradicting the hypothesis of standardization.

3. φ_t and ψ_t start in t_2. Then φ and ψ could start in different copies of t_2 in t^+. If φ_t and ψ_t are internal to t_2 then both $\partial_t^{w;\rho}(u) = \partial_t^{w;\sigma}(v)$ and $\partial_{t+}^{\rho}(u) = \partial_{t+}^{\sigma}(v)$ could hold, but in this case $(\mathbf{w};\rho)u$ and $(\mathbf{w};\sigma)v$ could be further simplified w.r.t. the extraction process, thus invalidating the hypothesis of canonicity. The last case is when φ_t and ψ_t exit from t_2. The unique possibility admitted by the requirement of canonicity is that they exit "from the top" (i.e. towards the application at w). This implies that φ_t and ψ_t must traverse the same bound edge of $\lambda x.\, t_1$ (for the same reason of the first subcase of 2). That is φ and ψ must start in the same copy of t_2. The contradiction $\partial_{t+}^{\rho}(u) = \partial_{t+}^{\sigma}(v)$ follows by iterating the reasoning. ■

7 Legal paths

We have proved in Section 5 that different degrees correspond to different paths in the original term. However, there are paths that are not associated with degrees. In this section we shall provide a complete characterization of paths yielded by degrees (*legal paths*), independently from the notion of labeling.

We start with providing the notion of *well balanced paths* (wbp's), that is a superclass of *legal paths*. We shall obtain legal paths by suitably constraining wbp's.

Definition 7.1 Let t be an expression. Any edge connecting the principal port of an application to an arbitrary operator, whose "type" ? may be λ, @ or v (variable), is a *well balanced path* (shortened into wbp) of type @-?.

Well balanced paths are then composed in the following way:

(λ-composition) Let ψ be a wbp of type @-v whose ending variable is bound by a λ-node $\mathbf{c}$ and φ be a wbp of type @-λ coming into $\mathbf{c}$. Then

$\psi \cdot (\varphi)^r \cdot u$ is a wbp, where u is the edge outgoing the second argument of the initial node of φ. The type of $\psi \cdot (\varphi)^r \cdot u$ depends on the node connected to u;

(@-composition) Let ψ be a wbp of type @-@ ending into a node **d** and φ be of type @-λ leading from **d** to some λ-node **c**. Then $\psi \cdot \varphi \cdot u$ is a wbp, where u outgoes **c** towards its body. The type of $\psi \cdot \varphi \cdot u$ depends on the connection of the edge u. ■

The idea is the following. Every wbp's of type @-λ corresponds to a "session" in the terminology of the introduction. Composition with a "session" explains how to resume a previous search according to the way we "entered" the session. Another way to understand wbp's is in relation with Proposition 5.3: every well balanced (sub-)path of type @-λ corresponds to an underlined or overlined label (see section 8), which must be surrounded by suitables atomic edges.

Example 7.2 Consider again the λ-term $\Delta\Delta$ (see the picture in the introduction). The path b is a wbp of type @-λ. d is a wbp of type @-v. By rule 1, $d(b)^r f = dbf$ is a wbp of type @-λ. It corresponds to the redex created after one step, along the unique derivation for $\Delta\Delta$. Now we can proceed. h is a wbp of type @-v, and we have already built the wbp dbf leading from an application to the binder of the ending variable of h. So we can apply again rule 1, obtaining the wbp $h(dbf)^r e = hfbde$ of type @-v. By a further application of rule 1, we finally get the wbp $hfbde(b)^r f = hfbdebf$ of type @-λ. This is the path associated to the unique redex created after two β-reductions.

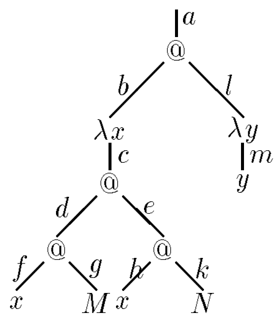
Up to now, we have a complete correspondence between "virtual" redexes and wbp's. Unfortunately, this is lost at the next step. Note first that we have now two @-λ paths leading to the same application marked with f in the original term, namely $\varphi = dbf$ and $\psi = hfbdebf$. By rule 1, we may build a @-v-path, $h(\psi)^r k$. The final variable is bound by the λ marked with f in the original term. So we may now proceed in two ways, according to the previous paths. If we follow φ, we end up with $h(\psi)^r k(\varphi)^r ebf$, that correctly correspond to the redex created after three β-reductions. But we may also repeat ψ an arbitrary number of times, building wbp's of the kind

$$h(\psi)^r k(\psi)^r k \cdots (\psi)^r k(\varphi)^r ebf$$

and no one of these paths is associated with a redex. ■

Let us provide a simpler example where the correspondence between wbp's and (virtual) redexes fails.

Example 7.3 Consider the λ-term $(\lambda x.(xM)(xN))(\lambda y.y)$, represented in the following picture:



We have two wbp's $\varphi = fbl$ and $\psi = hbl$ leading to the same λ. The two paths $d\varphi m(\varphi)^r g$ and $d\varphi m(\psi)^r k$ are both well balanced, but only the first one is "legal". ∎

So, wbp's are a superset of the set of paths yielded by degrees. This means that wbp's must be constrained by some proviso. Let us try to understand the problem. As we remarked in the introduction, the only non-determinism in the search algorithm (in the definition of wbp) is at the level of bound variables. If a bound variable v appears inside a wbp, it is eventually followed by a wbp φ of type @-λ (reversed) and an access to the argument N of the application. Now, suppose φ describes a cycle internal to N. Intuitively, we are working inside the instance of N individuated by the particular bound variable v. So, when we exit from N we "cannot jump" inside another instance, but we are forced to follow back the same path we used to access the argument.

We shall now formalize the previous intuition. The main problem is to capture the "right" notion of cycle, that will require an inductive definition. The basic case is when the cycle is *physically* internal to the argument N of the application. The problem is that we can exit from some free variable of N (free in N, but eventually bound in the initial term), make a cycle somewhere else, and come back again inside N.

Definition 7.4 Let φ be a wbp. An *elementary @-cycle* of φ is a subpath ψ starting from and ending to the argument edge p of a @-node (the negative auxiliary port), and internal to the argument N of the application (i.e. not traversing variables which are free in N). ∎

Let us come to the definition of cycle. For this purpose we shall also inductively define the notion of *v-cycle* (a cycle over a variable!).

Definition 7.5 Let φ be a wbp.

basic case Every elementary @-cycle is a @-cycle.

induction (v-cycles) Every cyclic path of the form $v\lambda(\varphi)^r@\psi@\varphi\lambda v$ where φ is a wbp and ψ is a @-cycle, is a v-cycle.

(@-cycles) Every path ψ starting from and ending to the argument edge p of a @-node (the negative auxiliary port), and composed of subpaths internal to the argument N with v-cycles over free variables of N is a @-cycle. ∎

The next step is to require that @-cycles must be surrounded by a unique common path to the "associated" λ. The unicity of such a λ is stated by the following proposition.

Proposition 7.6 Let φ be a wbp.

1. For every λ-node traversed by φ at its principal port there exists a unique decomposition of φ as $\zeta_1@\zeta_2\lambda\zeta_3$ or $\zeta_1\lambda(\zeta_2)^r@\zeta_3$, where $@\zeta_2\lambda$ is a wbp.
2. For every @-node traversed by φ, except the initial one, there exists a unique decomposition of φ as $\zeta_1@\zeta_2\lambda\zeta_3$ or $\zeta_1\lambda(\zeta_2)^r@\zeta_3$, where $@\zeta_2\lambda$ is a wbp. The initial @-node is joined with a λ-node if and only if the type of φ is @-λ.

Proof: Easy induction on the definition of wbp. ∎

Corollary 7.7 Let φ be a wbp with a @-cycle $@\,\psi\,@$. Then φ can be uniquely decomposed as

$$\zeta_1\,\lambda\,\zeta_2\,@\,\psi\,@\,(\zeta_3)^r\,\lambda\,\zeta_4$$

where both ζ_2 and ζ_3 are wbp's. ∎

In the situation of Corollary 7.7, we will say that ζ_2 and ζ_3 are the *call* and *return* paths of the @-cycle ψ. The last label of ζ_1 and the first label of ζ_4 will be named the *discriminants* of the call and return paths, respectively (note that discriminants are eventually edges relative to variables bound by λ).

Now, we are ready to state our *legality* condition for wbp's.

Definition 7.8 A wbp is a *legal path* if and only if the call and return paths of *any* @-cycle are one the reverse of the other and their discriminants are equal. ■

Example 7.9 Consider the two wbp's $dfblmlbfg$ and $dfblmlbhk$ of Example 7.3. In both cases we have an (elementary) @-cycle lml over the @-node labeled with @. In both cases the call and return path is b, but their discriminants are different in the second case (they are f and h). ■

8 Legal paths and redex families

In this section, we shall prove the bijective correspondence between legal paths and paths yielded by degrees.

Some preliminary definitions and results are required.

Definition 8.1 Let φ be a wbp over a labeled term t ($\mathtt{INIT}(t)$ is assumed). The label of φ is defined inductively as follows:

(basic case) the label of the edge marked by φ;

(@-composition) if $\varphi = @\varphi_1 @ \varphi_2 \lambda u$ then its label is $\ell_1 \overline{\ell_2} a$, where ℓ_1 is the label of φ_1, ℓ_2 is the label of φ_2 and a is the label of the edge u;

(λ-composition) if $\varphi = @\varphi_1 \lambda \varphi_2 @ u$ then its label is $\ell_1 \underline{\ell_2} a$, where ℓ_1 is the label of φ_1, ℓ_2 is the label of φ_2 and a is the label of the edge u. ■

According to the above definition, it is immediate to verify that, if ℓ is the label of a wbp φ, then $\varphi = \mathtt{path}(\ell)$.

Let $t \xrightarrow{u} t'$. Every (arbitrary) path in t' has an *ancestor* in t. Its definition is pretty intuitive, so we shall not be pedantic, here. In particular, we shall just define the notion of ancestor for single edges in t'. This is extended to paths by composition, in the obvious way.

Suppose that $t = C[(\lambda x.M)N]$ and $t' = C[M[^N/_x]]$, where $C[\,]$ is some context. Note first that some edges in t' are "residuals" of edges in t. This is the case for every edge v' internal to M, to some instance of N, or belonging to the context. If v' is a residual of v in the previous sense, then v is the origin of v'. The problem is when v' is a new connection created by firing the redex u. Let @ and λ be the two nodes connected by u in t. Three cases are possible:

- v' is a connection between the context and the body M. The ancestor of v' is $a \cdot u \cdot b$, where a is the edge leading from the context to the positive port of @, and b is the edge leading from λ to M.

- v' is a connection between M and the i-th instance of N. The ancestor of v' is $b \cdot (u)^r \cdot c$, where b is the edge leading from M to the i-th instance of the variable bound by λ, and c is the edge leading from @ to N.

- v' is a connection between the context and N. This is only possible when $M = x$, and it is an obvious combination of the previous cases. In particular, the ancestor of v' is $a \cdot u \cdot b \cdot (u)^r \cdot c$, with a, b and c as above.

We leave to the reader the care to generalize the definition of ancestor from edges to arbitrary paths (we have to prove that we preserve the connections, but this is easy).

The definition of ancestor is then extended to derivations in the obvious way.

Proposition 8.2 Every ancestor of a wbp is still a wbp. ■

Lemma 8.3 Let $t \xrightarrow{u} t'$, and φ be the ancestor of a *legal* path φ' in t'. Then

1. Every @-cycle in φ relative to an application $\mathbf{d}$ not fired by u is ancestor of an @-cycle in φ' over any residual of $\mathbf{d}$.

2. Every v-cycle in φ relative to an instance x of a variable not bound by the λ in u is ancestor of an v-cycle in φ' over any residual of x.

Proof: The proof is by induction on the definition of cycles.

- In case the @-cycle is elementary, property 1 is trivial.

- Let us consider the inductive case.

(v-cycles) Let $v\lambda(\varphi_1)^r @\psi @\varphi_1 \lambda v$ be the cycle. Since v is not bound by the λ in u, the @-cycle ψ cannot be relative to the application fired by u (indeed, u is the unique wbp in t relative to this application). So we may apply the induction hypothesis, and we know that there exists a cycle ψ' inside φ' whose ancestor is ψ. Recall that φ' is legal, so we must found some subpath of the shape $v'\lambda(\varphi'_1)^r @\psi' @\varphi'_1 \lambda v'$ containing ψ'. Since the ancestor of a wbp is still a wbp, φ_1 must eventually be the ancestor of φ'_1, and similarly for v' and v.

(@-cycles) Consider a @-cycle ψ inside φ and relative to an application $\mathbf{d}$ in t. Let ψ' be the sub-path of φ' whose ancestor is ψ. This eventually exists by the edge-wise definition of ancestors. Consider all the "maximal" v-cycles (i.e. v-cycles not contained in other v-cycles) in ψ not relative to variables bound by the λ in u. By induction, these are ancestor of v-cycles in ψ'. Moreover these are all the possible maximal v-cycles in ψ' since variables bound by the lambda in u disappear after the reduction. The remaining portion of paths in ψ' is eventually inside the argument of the unique residual $\mathbf{d}'$ of $\mathbf{d}$ (by the connection of ψ' and a simple case inspection on the position of $\mathbf{d}$ w.r.t. u). So ψ' is indeed a @-cycle over $\mathbf{d}'$. ■

Lemma 8.4 Let $t \xrightarrow{u} t'$, and φ be the ancestor of a *legal* path φ' in t'. Then every @-cycle ψ in φ relative to an application $\mathbf{d}$ not fired by u is legal.

Proof: By Lemma 8.3, ψ is ancestor of a @-cycle ψ' in φ'. Since φ' is legal we must have in φ' some subpath of the kind $v'\lambda(\varphi'_1)^r @\psi' @\varphi'_1 \lambda v'$ containing ψ'. So the structure of φ around the @-cycle ψ has the shape $v\lambda(\varphi_1)^r @\psi @\varphi_1 \lambda v$, where φ_1 is the ancestor of φ'_1, and v' is the ancestor of v. ■

Lemma 8.5 Let $t \xrightarrow{u} t'$, and φ be the ancestor of a *legal* path φ' in t'. Then every @-cycle ψ in φ relative to the application $\mathbf{d}$ fired by u is legal.

Proof: The call and return paths are obviously equal (they are u). We must only prove that also the discriminants are equal. The proof is by induction on the definition of @-cycle. The basic case is easy. Let us consider the inductive case. The path is then composed by subpaths internal to the argument N of the redex u and v-cycles over free variables of N. By lemma 8.3 these v-cycles are ancestors of v-cycles in φ'. Consider such a cycle. It will be relative to some free variable of some instance N_i of N in t. Note now that all instances of N are disjoint in t'. Since the path φ' is connected, it must eventually define the "cycle" originating ψ inside this instance N_i, that ensures the equality of discriminants. ■

Proposition 8.6 The ancestor of a legal path φ' in t' along $t \xrightarrow{u} t'$ is a legal path φ in t.

Proof: Easy consequence of Lemmas 8.3 and 8.4. ■

Proposition 8.7 Labeling of paths is unchanged in ancestors. ■

Theorem 8.8 Every path yielded by degrees of redexes is a legal path.

Proof: By Proposition 8.6, and the fact that if ℓ is the label of a path φ, then $\varphi = \mathtt{path}(\ell)$. ■

The strategy to show the vice versa of Theorem 8.8 is based on the proof that legal paths can be uniquely "contracted" to legal paths by firing the leftmost outermost redex. A preliminary immediate result is required.

Proposition 8.9 Let $u = (\lambda x.M)N$ be a redex in t, and let φ, be a legal path internal to N. Let $t \xrightarrow{u} t'$. Then there exists a legal path φ_i in each instance of N_i whose ancestor is φ. ■

Proposition 8.10 Let φ, $|\varphi| > 1$, be a legal path in t and let $u = (\lambda x.M)N$ be the leftmost outermost redex traversed by φ in t. Let $t \xrightarrow{u} t'$. Then there exists a *unique* legal path φ' in t' whose ancestor is φ.

Proof: (Sketch) We start proving that there exist a *unique* wbp φ' in t' whose ancestor is φ (the only problem is the connection of φ'). We work by induction on the structure of wbp of φ.

The basic case is vacuous. For simplicity we shall only consider the case of @-composition. Thus, let then $\varphi = @_1\varphi_1@_2\varphi_2\lambda_1$. If φ_1 (φ_2) is internal to N, by Proposition 8.9, we have a "copy" of φ_1 (φ_2) inside every instance of N, and we are free to choose the only one that matches the "contractum" of φ_2 (φ_1). Suppose otherwise. By induction we know that there are two paths $@'_1\varphi'_1@'_2$ and $@''_2\varphi'_2\lambda'_1$ whose ancestors are φ_1 and φ_2, respectively. $@'_2$ and $@''_2$ are both residuals of $@_2$, but, a priori, nothing ensures that $@'_2 = @''_2$. The only problematic case is when the composition is internal to the argument N. Recall that both paths φ_1 and φ_2 exit from N. Since u is the leftmost outermost redex, they must eventually exit from the "top", i.e. we have an access to N. In particular, we have a final sub-path $u\zeta_1@_1$ of φ_1, where ζ_1 is internal to N. Similarly we must have an initial sub-path $@_1\zeta_2 u$ of φ_2 with ζ_2 internal to N. So the connected path $\zeta_1\zeta_2$ is a (elementary) cycle for the application in u. Since φ is legal, the discriminants for the two occurrences of u must be the same. So $@'_2 = @''_2$, since they will be in the same instance of N. Therefore φ' is a wbp.

We must still prove that φ' is legal. This is done by following the same ideas as for Lemmas 8.3 and 8.4. Namely, we have that the unique contractum in φ' of a @-cycle in φ not relative to the application fired by u is still a @-cycle, and similarly for a v-cycle not relative to a variable bound by the λ fired by u. Legality easily follows, by the unicity of the contractum.

Theorem 8.11 If φ is a legal path, there exists a degree ℓ such that $\mathtt{path}(\ell) = \varphi$.

Proof: By Proposition 8.10 we can "contract" a legal path into another legal path by firing the leftmost outermost redex. The length of the contractum is strictly shorter than that of its ancestor, and we eventually end up with a single redex. Since labels are preserved by contraction, the path yielded by the ending redex is exactly φ. ■

Proposition 8.12 The canonical derivation corresponding to the label of a legal path φ is obtained by firing the leftmost-outermost redex in φ, and iterating this process over its unique residual.

Proof: By induction on the length of the legal path. The basic case is obvious. Then take a legal path φ. By Lemma 6.2 the leftmost-outermost redex traversed by φ is the first-one fired by the canonical derivation yielding the label of φ. Let it be $\mathbf{u}$. By Proposition 8.10 the residual of φ w.r.t. $\mathbf{u}$ is unique and it is legal. Let it be φ' (notice that its length is shorter than φ). By induction, φ' has a canonical derivation fitting with the statement. Let it be σ. The derivation $\mathbf{u};\sigma$ is canonical, by the definition of the extraction relation, and the fact that $\mathbf{u}$ is needed. Since the derivation $\mathbf{u};\sigma$ preserves the label of (all residuals of) φ, the proposition is proved.

Note also that the canonical derivation yielding a redex labeled as φ is unique, for the bijective correspondence between canonical derivations and degrees. ■

The previous proposition essentially states the intuitive fact that *all and only* the ("virtual") redexes along a legal path are needed for its contraction to a single redex. The unicity of the residual of a legal path w.r.t. its leftmost outermost redex provides a unique, standard (and thus canonical) way to perform this contraction.

9 Legal paths and optimal reductions

The notion of legal path helps both in the simplification of the theory of optimality and in the understanding of optimal evaluators for the λ calculus. For instance, using paths, we have provided a new, simple proof of the coincidence of labeling and extraction (see section 6). As far as the optimal evaluator is concerned, due to Theorem 8.11, the implementation must always have a unique representation of every legal path. In particular, legal paths must be *physically* shared. The evaluation proceeds by contracting legal paths, provided that every contractum of a legal subpath has a unique representation.

More precisely, suppose to have an explicit operator of duplication (a *fan*, in Lamping's terminology). Consider a legal path φ. An essential condition to get optimality is that a *fan* external to φ can never enter the path. On the other side, a *fan* already internal to φ can be freely moved along φ (provided it does not enter subpaths of type @-λ). That is, we may pursue the duplication *inside* a path @-λ, since the portion of paths we are duplicating eventually belong to different legal paths. As a matter of fact, a prerequisite chain [10] for @ is always a prefix of a legal path, up to the first atomic edge representing a redex. The interesting fact is that this prefix is always *unique* for every legal path starting from a same node. In a sense, prerequisite chains are the deterministic part of legal paths. Similarly, a prerequisite chain of a λ-node is the *unique* common suffix of every legal path starting form the last edge representing a redex.

There is a strong relation between our legal paths and Lamping-Gonthier's *consistent paths*. These are used to read back λ-expressions from sharing graphs. In [4], it is proved that legal paths are exactly wbp's that are consistent w.r.t. the *context semantics* [10, 7] yielding a further, alternative characterization of legal paths. In the same paper, it is also fixed the correspondence between Danos and Regnier's paths [5] and legal paths. We recall that the former ones have a strong geometric motivation (in the sense of Girard's "Geometry of Interaction") and a very interesting algebraic flavour. The equivalence of all these different notions of path clarify the relation between optimal reductions, Linear Logic and the geometry of interaction, already pointed out in [7, 8].

10 Conclusions

We have provided a new characterization of the family relation in the λ calculus in terms of suitable paths in the initial term of the derivation. In, particular, we have proved that two redexes in a same family comes from the contraction of a unique common path in the initial term, yielding new evidence to their "sharable" nature. From this respect, our work sheds a new light over graph reduction techniques for optimal implementations, elaborating Danos and Regnier's idea of a computation as a *travel along a path* [14].

From the theoretical point of view, legal paths help to understand Levy's labels, providing a new insight in the algebraic structure of degrees. Moreover, in our experience, legal paths are the most friendly and intuitive tool for reasoning about redex-families, leading to a significant simplification of many results concerning them.

Acknowledgements We have enjoyed discussions with Gerard Boudol, Georges Gonthier and Jean Jacques Lévy.

References

[1] A. Asperti. Linear logic, comonads, and optimal reductions. To appear in Fundamenta Informaticae, special issue devoted to "Categories in Computer Science".

[2] A. Asperti and C. Laneve. Interaction Systems 1: The theory of optimal reductions. Technical Report 1748, INRIA-Rocquencourt, September 1992.

[3] A. Asperti and C. Laneve. Optimal reductions in Interaction Systems. In *TapSoft '93*, Lecture Notes in Computer Science. Springer-Verlag, 1993.

[4] A. Asperti, V. Danos, C. Laneve and L. Regnier. Paths in the λ-calculus. In *Proceedings* 9^{th} *Annual Symposium on Logic in Computer Science,* Paris, 1994.

[5] V. Danos and L. Regnier. Local and asynchronous beta-reduction. In *Proceedings* 8^{th} *Annual Symposium on Logic in Computer Science,* Montreal, 1993.

[6] J. Field. On laziness and optimality in lambda interpreters: tools for specification and analysis. In *Proceedings* 17^{th} *ACM Symposium on Principles of Programming Languages*, pages 1 – 15, 1990.

[7] G. Gonthier, M. Abadi, and J.J. Lévy. The geometry of optimal lambda reduction. In *Proceedings* 19^{th} *ACM Symposium on Principles of Programmining Languages*, pages 15 – 26, 1992.

[8] G. Gonthier, M. Abadi, and J.J. Lévy. Linear logic without boxes. In *Proceedings* 7^{th} *Annual Symposium on Logic in Computer Science*, 1992.

[9] Y. Lafont. Interaction Nets. In *Proceedings* 17^{th} *ACM Symposium on Principles of Programmining Languages*, pages 95 – 108, 1990.

[10] J. Lamping. An algorithm for optimal lambda calculus reductions. In *Proceedings* 17^{th} *ACM Symposium on Principles of Programmining Languages*, pages 16 – 30, 1990.

[11] C. Laneve. *Optimality and Concurrency in Interaction Systems*. PhD thesis, Technical Report TD – 8/93, Dip. Informatica, Università di Pisa, March 1993.

[12] J.J. Lévy. *Réductions correctes et optimales dans le lambda calcul*. PhD thesis, Université Paris VII, 1978.

[13] J.J. 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*, pages 159 – 191. Academic Press, 1980.

[14] L. Regnier. Lambda Calcul et Réseaux. Thèse de doctorat, Université Paris VII. 1992.