Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science 171 (2007) 111–126
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

A Fully Labelled Lambda Calculus: Towards Closed Reduction in the Geometry of Interaction Machine

Nikolaos Siafakas[1]

Department of Computer Science
King's College London, UK

Abstract

We investigate the possibility of performing new reduction strategies with the Geometry of Interaction Machine (GOIm). To this purpose, we appeal to Lévy's labelled lambda calculus whose labels describe: a) the path that the GOIm will follow in the graph of a term and b) the operations that the GOIm requires to compute the multiplicative part from the Multiplicative and Exponential Linear Logic encoding that the machine uses. Our goal is to unveil the missing exponential information in the structure of the labels. This will provide us with a tool to talk about strategies for computing paths with the GOIm.

Keywords: Lambda calculus, Labels, Paths, Geometry Of Interaction

1 Introduction

There is a well established connection between labels in Lévy's labelled λ-calculus and paths in the graph of a term [1,4,2]. If we compute the normal form of a term in the labelled λ-calculus, then the resulting label will describe a path in the graph of the term. The Geometry Of Interaction Machine [12], which is an implementation of Girard's Geometry of Interaction semantics for Linear Logic [10], will follow exactly the path induced by the label. The investigation of the structure of the labels allowed the identification of the call-return symmetry [4], which has led to optimisations [6,7] where the length of the path to be traversed is significantly reduced. However:

- The structure of the labels is different from the structure of paths: labels talk about redex contractions whereas paths in the GOIm about the dynamics of Linear Logic cut elimination.

[1] email:nikolaos.siafakas@kcl.ac.uk

doi:10.1016/j.entcs.2006.12.040

- The structure of the paths depends on the choice of translation of the λ-calculus into Linear Logic proof nets (call-by-value or call-by-name).
- Many equivalences are known to date but properties found in the labels have to be transposed to paths and vice versa.

Our goal is to identify structure in the labels that will allow us to reason about new ways for computing paths with the GOIm. Therefore, we investigate a new set of labels, which corresponds closer to the paths.

2 Background and Motivation

In this section we set up the conventions that we will use in the rest of the paper. Our point of departure is Lévy's labelled λ calculus [11] whose labels give a simple notion for a "path in a term".

Definition 2.1 The set of labels is formed by the following grammar:

$$\alpha, \beta := a \mid \alpha\beta \mid \overline{\alpha} \mid \underline{\alpha}$$

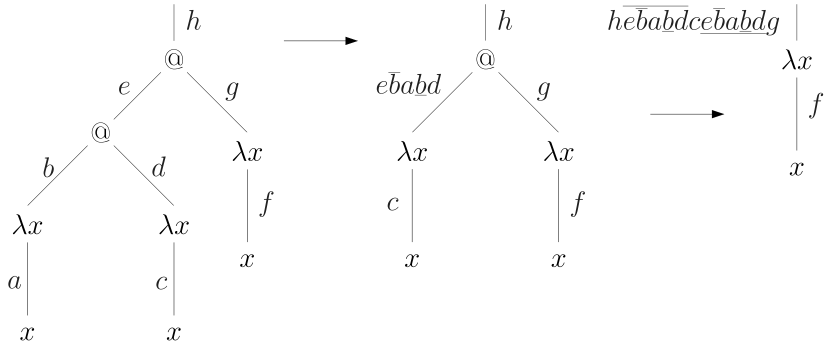
where (Latin) a is an atomic label. Labelled terms are terms of the λ-calculus where each sub-term T has a label attached on it: T^α. Subterms in the initial form of a term receive a distinct atomic label. Labelled β-reduction is given by

$$((\lambda x.M)^\alpha N)^\beta \to \beta\overline{\alpha} \bullet M[\underline{\alpha} \bullet N/x]$$

where $\bullet$ concatenates labels with labelled terms, defined by $\beta \bullet T^\alpha = T^{\beta\alpha}$. Substitution is implicit and operates as follows

$$\begin{aligned}
x^\alpha[N/x] &= \alpha \bullet N \\
y^\alpha[N/x] &= y^\alpha \\
(\lambda y.M)^\alpha[N/x] &= (\lambda y.M[N/x])^\alpha \\
(MN)^\alpha[P/x] &= (M[P/x]N[P/x])^\alpha
\end{aligned}$$

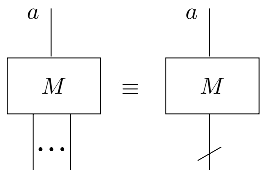
Example 2.2 The term III, $I = (\lambda x.x)$ reduces as follows:



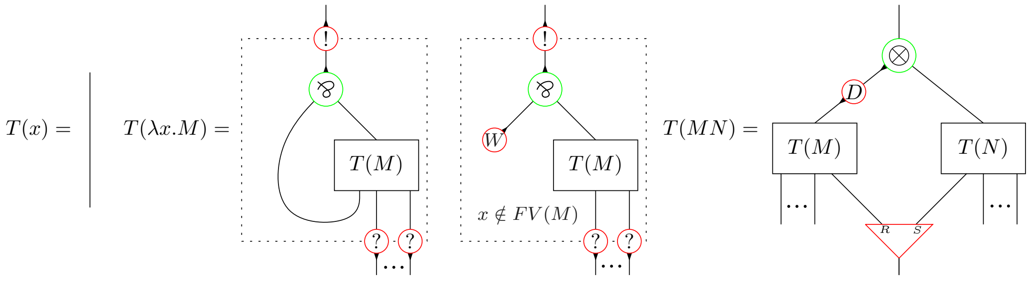
If we reverse the underlines then the resulting label induces a path that the GOIm will compute without reducing the term. However, the GOIm does not act on syntax trees of the λ-calculus but on graphs, which are translations of the

λ-calculus into Linear Logic proof-nets [9]. Computation then follows the idea of a token traversing the edges of the graph. Here we provide the call-by-value translation $T(\cdot)$ with which we will work throughout this paper [12].

Definition 2.3 The general form of a term is given by



where we attach an atomic label to the root of each sub-term. Edges at the bottom of the structure denote the free variables of the term. The translation is given by the following recursive definition:



Some remarks are in order: In $T(\lambda x.M)$, we assume that the bound variable x is at the leftmost edge of the free variables of $T(M)$. We attach a node W if $x \notin \mathsf{fv}(M)$. Intuitively, the box structure can be thought of as “the scope” of a function. In terms of Linear Logic, it is a net that can be used in a non-linear way. The graphs presented here are proof nets, which are oriented in a way that closely resembles λ-calculus syntax trees. Notice that axiom and cut links are hidden within the structures. We distinguish between two kinds of nodes:

(i) Multiplicative nodes: Tensor ($\otimes$), Par($\parr$)

(ii) Exponential nodes: Of Course (!), Dereliction (D), Weakening (W), Why Not (?), Contraction (R, S)

The arrows on the nodes are not standard proof net notation but we place them in order to keep track of the original orientation of the nets. This idea is borrowed from Interaction Net notation where arrows are used to indicate principal ports of nets. We will not require the full power of MELL cut elimination but only Closed MELL cut elimination, which we present in Figure 1 with the original orientation of the nets. We refer the reader to [13] for a more formal treatment.

The overlining and underlining in Lévy’s labelled λ-calculus captures redexes that have been contracted during reduction. In other words, these labels mark the points of contact of the (virtual) redexes in a term. If we transpose this information to proof-nets then overlining and underlining hints about where the multiplicative nodes are located in the labels. This is also the case for the call-by-name translation, which we do not consider here.

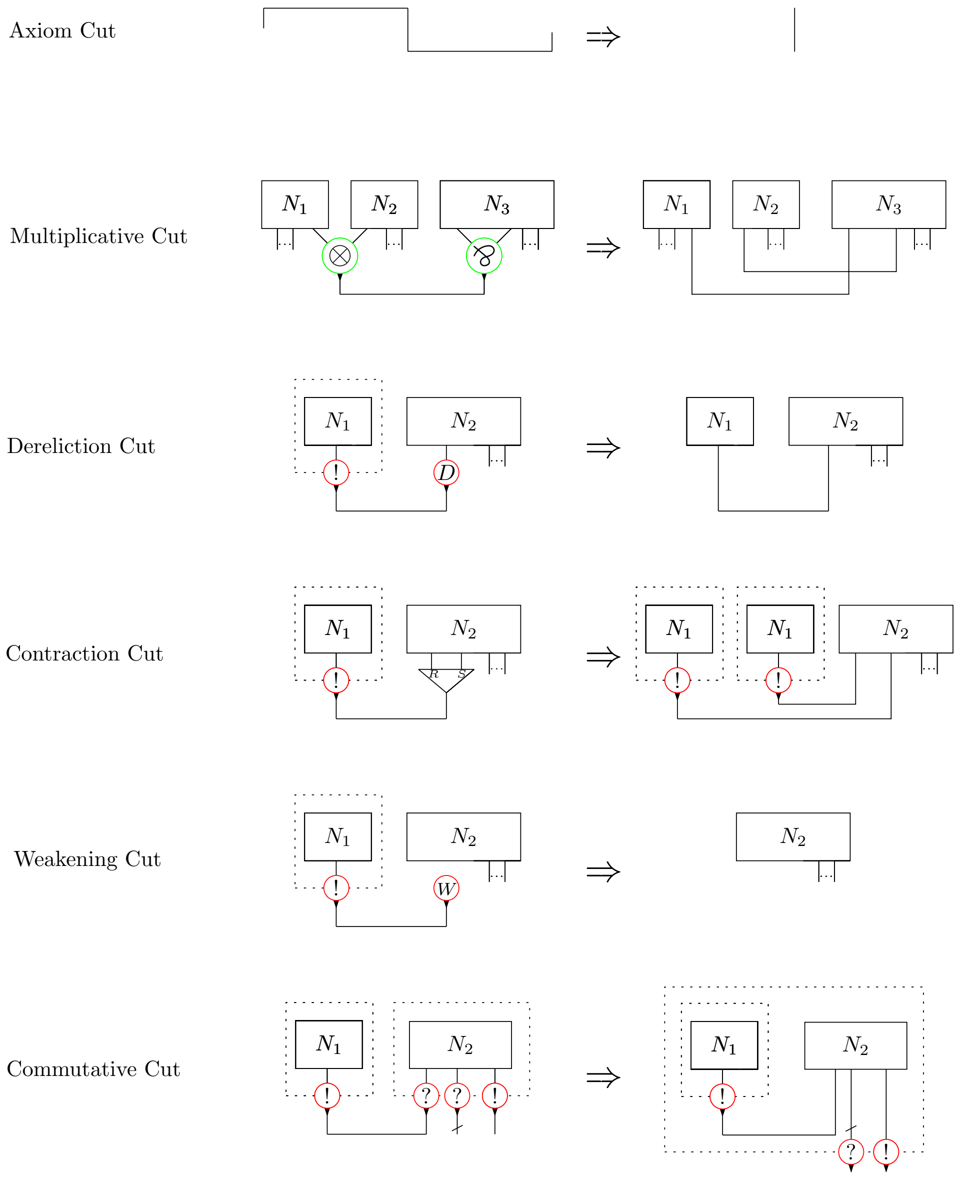


Fig. 1. Closed Cut Elimination

In the next section we show that it is possible to add the missing exponential information during labelled reduction where we make use of a weak, α-conversion free λ-calculus. A weak λ-calculus is also studied in [5], which has been related to Wadsworth's graph representations. In section 4 we relate the structure obtained from the labels with Linear Logic cut elimination where we make the notion of "connectivity" important (see for instance [3]). In section 5 we comment on how the extra structure helps in understanding and exploring new strategies for the

GOIm.

3 Paths in the calculus of closed functions

Our approach is similar to Lévy's labelled reduction where we attach labels to terms and capture information during reduction. The labelled β-rule can provide some information about the location of the exponential nodes. In the call-by-value translation, it is trivial to identify the location of D-nodes and !-nodes since these live beneath the multiplicative nodes. This is because the translation of a redex involves multiplicative nodes, which are intercepted by a D and a !-node. Unfortunately, the β-reduction step cannot help further; we have to focus on the substitutions in order to identify the locations of the remaining exponential nodes. However, substitutions are propagated exhaustively and in an uncontrolled way. For instance, it is known that paths can be copied but this is not resembled by copying the substitution in:

$$(MN)^{\alpha}[P/x] = (M[P/x]N[P/x])^{\alpha}$$

The explicit substitution calculi in [8] overcome these deficiencies. In this section we provide a labelled version of the "calculus of closed functions" (λ_{cf}), which will allow us to obtain explicit paths for the call-by-value translation.

Definition 3.1 Terms in the λ_{cf}-calculus are λ-terms with explicit constructs for copying (δ) and erasing (ϵ) and are presented along with their variable constraints.

Term	Variable Constraint	Free variables
x	-	$\{x\}$
$\lambda x.M$	$x \in \mathsf{fv}(M)$	$\mathsf{fv}(M) - \{x\}$
MN	$\mathsf{fv}(M) \cap \mathsf{fv}(N) = \emptyset$	$\mathsf{fv}(M) \cup \mathsf{fv}(N)$
$\epsilon_x.M$	$x \notin \mathsf{fv}(M)$	$\mathsf{fv}(M) \cup \{x\}$
$\delta_x^{y,z}.M$	$x \notin \mathsf{fv}(M), y \neq z, \{y,z\} \subseteq \mathsf{fv}(M)$	$(\mathsf{fv}(M) - \{y,z\}) \cup \{x\}$
$M[N/x]$	$x \in \mathsf{fv}(M), (\mathsf{fv}(M) - \{x\}) \cap \mathsf{fv}(N) = \emptyset$	$(\mathsf{fv}(M) - \{x\}) \cup \mathsf{fv}(N)$

Definition 3.2 Lambda terms are compiled into this set of terms (λ_c) via the translation $\langle\!\langle \cdot \rangle\!\rangle$. We assume compilation of closed terms:

$$
\begin{aligned}
\langle x \rangle &= x \\
\langle MN \rangle &= \langle M \rangle \langle N \rangle \\
\langle \lambda x.M \rangle &= \lambda x.[\![x]\!]\langle M \rangle \text{ if } x \in \mathsf{fv}(M) \\
&= \lambda x.\epsilon_x.\langle M \rangle \text{ otherwise}
\end{aligned}
$$

We define $[\![\cdot]\!]$:

$$
\begin{aligned}
[\![x]\!]x &= x \\
[\![x]\!](\lambda y.M) &= \lambda y.[\![x]\!]M \\
[\![x]\!](MN) &= \delta_x^{y,z}.[\![y]\!](M[x := y])[\![z]\!](N[x := z]) && x \text{ free in } M \text{ and } N \ (y,\, z \text{ fresh}) \\
&= ([\![x]\!]M)N && x \text{ free in } M \text{ only} \\
&= M([\![x]\!]N) && x \text{ free in } N \text{ only} \\
[\![x]\!](\epsilon_y.M) &= \epsilon_y.[\![x]\!]M \\
[\![x]\!](\delta_y^{u,v}.M) &= \delta_y^{u,v}.[\![x]\!]M
\end{aligned}
$$

Example 3.3 Here we provide some example terms that are yielded by the compilation:

- $\langle \lambda x.\lambda y.x \rangle = \lambda x.\lambda y.\epsilon_y.x$
- $\langle (\lambda x.xx)(\lambda x.x) \rangle = (\lambda x.\delta_x^{y,z}.yz)(\lambda x.x)$

Notice that the compilation does not apply to open terms. For instance,

$$\langle (\lambda x.x)(yy) \rangle = (\lambda x.x)(yy)$$

but the translated application (yy) does not satisfy the variable constraints of λ_c-terms. We refer the reader to [8] for the full version of the compilation.

Definition 3.4 Labelled terms are λ_c-terms where every sub-term T has a label: T^α. The set of labels is defined by the following grammar:

$$\alpha, \beta := a \mid \alpha\beta \mid \overline{\alpha} \mid \underline{\alpha} \mid C; \qquad C := \overrightarrow{E} \mid \overleftarrow{E}; \qquad E := D \mid ! \mid ? \mid R \mid S \mid W;$$

This is the same set as the set of Lévy's labels where we add (atomic) markers for exponential nodes, each of which has an associated direction. Multiplicative information is kept implicit via overlining and underlining.

Definition 3.5 The Beta rule of the labelled calculus λ_{lcf} is defined by

$$\textsc{Beta}: \qquad ((\lambda x.M)^\alpha N)^\beta \rightarrow_{\lambda_{lcf}} \beta\overline{\overrightarrow{\mathsf{D}}\alpha\overleftarrow{!}} \bullet M[\underline{(\overrightarrow{\mathsf{D}}\alpha\overleftarrow{!})^r} \bullet N/x]$$

where we impose the condition that $\mathsf{fv}((\lambda x.M)^\alpha) = \varnothing$. The operator $\bullet$ and the function $(\cdot)^r$ which reverses labels are defined as follows::

$$\begin{array}{llll}
(a)^r &= a & \beta \bullet x^{\alpha} &= x^{\beta\alpha} \\
(\alpha\beta)^r &= (\beta)^r \cdot (\alpha)^r & \beta \bullet (\lambda x.M)^{\alpha} &= (\lambda x.M)^{\beta\alpha} \\
(\overline{\alpha})^r &= \overline{(\alpha)^r} & \beta \bullet (MN)^{\alpha} &= (MN)^{\beta\alpha} \\
(\underline{\alpha})^r &= \underline{(\alpha)^r} & \alpha \bullet (\delta_x^{y\,z}.M) &= (\delta_x^{y\,z}.\alpha \bullet M) \\
(\overrightarrow{E})^r &= \overleftarrow{E} & \alpha \bullet (\epsilon_x.M) &= (\epsilon_x.\alpha \bullet M) \\
(\overleftarrow{E})^r &= \overrightarrow{E} & &
\end{array}$$

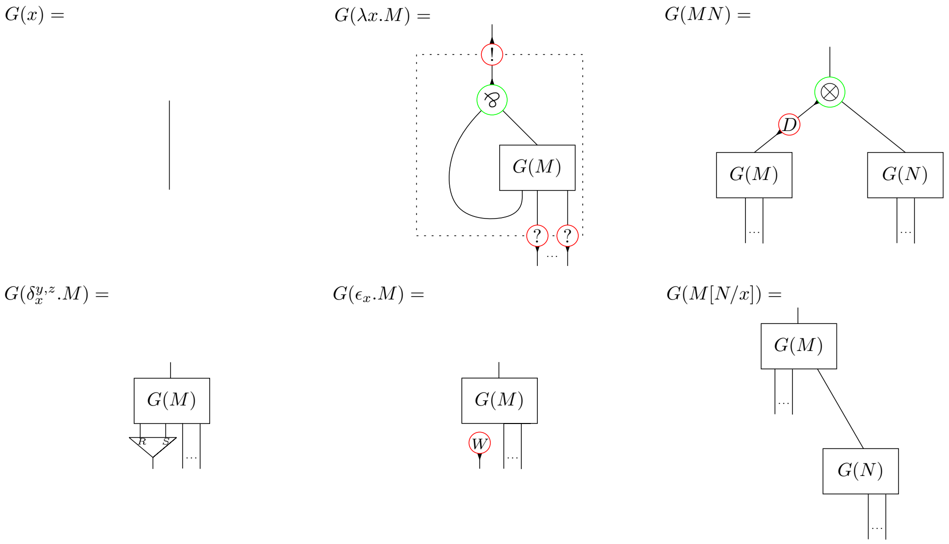
Substitution rules (σ) are placed at the same level as the BETA rule and are presented below.

Rule	Reduction			Condition
LAM	$(\lambda y.M)^{\alpha}[N/x]$	$\rightarrow_{\lambda_{lcf}}$	$(\lambda y.M[\overrightarrow{?} \bullet N/x])^{\alpha}$	$\mathsf{fv}(N) = \varnothing$
APP1	$(MN)^{\alpha}[P/x]$	$\rightarrow_{\lambda_{lcf}}$	$(M[P/x]N)^{\alpha}$	$x \in \mathsf{fv}(M)$
APP2	$(MN)^{\alpha}[P/x]$	$\rightarrow_{\lambda_{lcf}}$	$(MN[P/x])^{\alpha}$	$x \in \mathsf{fv}(N)$
CPY1	$(\delta_x^{y\,z}.M)[N/x]$	$\rightarrow_{\lambda_{lcf}}$	$M[\overrightarrow{\mathsf{R}} \bullet N/x][\overrightarrow{\mathsf{S}} \bullet N/x]$	$\mathsf{fv}(N) = \varnothing$
CPY2	$(\delta_x^{y\,z}.M)[N/x']$	$\rightarrow_{\lambda_{lcf}}$	$(\delta_x^{y\,z}.M[N/x'])$	~
ERS1	$(\epsilon_x.M)[N/x]$	$\rightarrow_{\lambda_{lcf}}$	$M, \quad \{\overrightarrow{\mathsf{W}} \bullet N\} \cup B$	$\mathsf{fv}(N) = \varnothing$
ERS2	$(\epsilon_x.M)[N/x']$	$\rightarrow_{\lambda_{lcf}}$	$(\epsilon_x.M[N/x'])$	~
VAR	$x^{\alpha}[N/x]$	$\rightarrow_{\lambda_{lcf}}$	$\alpha \bullet N$	~
CMP	$M[P/y][N/x]$	$\rightarrow_{\lambda_{lcf}}$	$M[P[N/x]/y]$	$x \in \mathsf{fv}(P)$

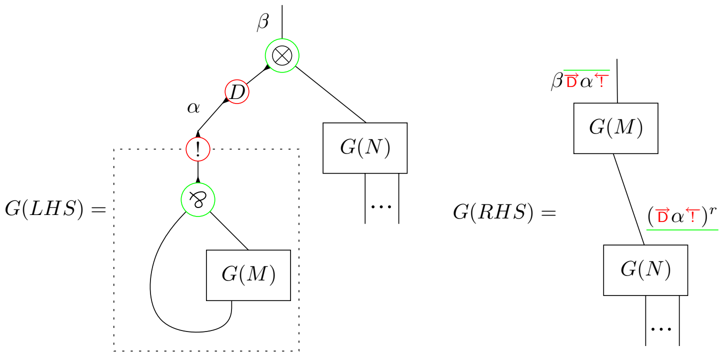
Remark 3.6 The only rule that creates a substitution is the BETA rule. The rule LAM captures the situation where the substitution has to leave the function scope of the abstraction in which the substitution has been originally created and enters a sub-function scope. The controlled copying and erasing (CPY1 and ERS1) of substitutions allows the identification of paths that start from contraction nodes and weakening nodes respectively. Erased paths are kept in a set B. Notice that explicit labelling on copying (δ) and erasing (ϵ) constructs is omitted: these are used just to guide the substitutions. We inherit a number of properties from λ_{cf}: our calculus is α-conversion free and closed substitutions do not remain blocked. On the other hand, the calculus is weak but is adequate for the evaluation of programs (closed terms).

A closer look at how the calculus operates will be presented via the graphical representation of the rewrite rules. For this reason, we define a translation for λ_c

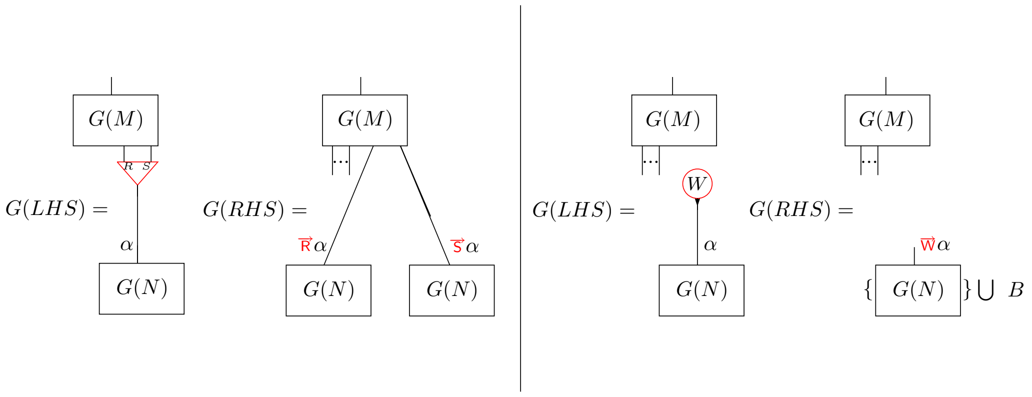
terms into Linear Logic proof nets, which we present below:



Example 3.7 Here is the graphical representation of the left and right hand sides of the Beta rule. There are two cut elimination steps involved in order to achieve this rewrite: 1) a closed dereliction cut and 2) a multiplicative cut.


$G(LHS) =$
β
α
D
$G(M)$
$G(N)$
$G(RHS) =$
$\beta\overline{\vec{D}\alpha\overleftarrow{T}}$
$\underline{(\vec{D}\alpha\overleftarrow{T})^r}$

The rules Cpy1 and Ers1 behave as follows:



Remark 3.8 There are many paths in a program and the task of the GOIm is to

find the path that survives the process of reduction. In other words, there exist erased sub-paths that are not part of the normal form of a program. However, when studying strategies for the GOIm, it is of interest to identify paths that lead to discarded arguments. Since substitutions are discarded in a controlled way, we keep these paths in the set B.

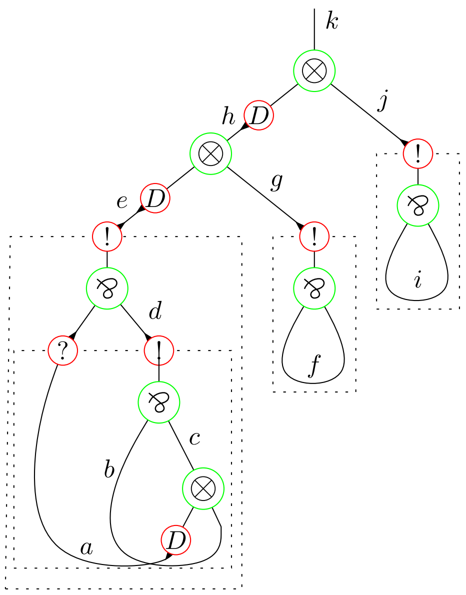
Example 3.9 We provide an example path yielded by the reduction of $(\lambda xy.xy)II$, $I = (\lambda x.x)$.

$$(((\lambda x.(\lambda y.(x^a y^b)^c)^d)^e(\lambda x.x^f)^g)^h(\lambda x.x^i)^j)^k \to^*_{\lambda_{lcf}} (\lambda x.x^i)^\phi$$

where

$$\phi = k\,\overline{\overrightarrow{\mathsf{D}}\,h\,\overline{\overrightarrow{\mathsf{D}}\,e\,\overleftarrow{!}}\,d\,\overleftarrow{!}}\,c\,\overline{\overrightarrow{\mathsf{D}}\,a\,\overrightarrow{?}\,\underline{\overrightarrow{!}\,e\,\overleftarrow{\mathsf{D}}}\,g\,\overleftarrow{!}}\,f\,\underline{\overrightarrow{!}\,g\,\underline{\overrightarrow{\mathsf{D}}\,e\,\overleftarrow{!}}\,\overleftarrow{?}\,a\,\overleftarrow{\mathsf{D}}}\,b\,\underline{\overrightarrow{!}\,d\,\overline{\overrightarrow{!}\,e\,\overleftarrow{\mathsf{D}}}\,h\,\overleftarrow{\mathsf{D}}}\,j$$

The corresponding graph is presented below:



In the remainder of this section we show that the labelled case preserves confluence. The proofs presented here are an adaption from [8] to the labelled case.

Lemma 3.10 *For $\bullet$ and substitution we have $(\beta \bullet M)[N/x] = \beta \bullet (M[N/x])$.*

Proof. By induction on the structure of M. We show the case for δ-terms:
$M = (\delta^{y\,z}_x.P)$. The lhs is

$$\begin{aligned}(\beta \bullet (\delta^{y\,z}_x.P))[N/x] \quad &= (\delta^{y\,z}_x.(\beta \bullet P))[N/x] \text{ by definition of } \bullet \\ &\to_{\lambda_{lcf}} (\beta \bullet P)[\overrightarrow{\mathsf{R}} \bullet N/y][\overrightarrow{\mathsf{S}} \bullet N/z] \\ &= \beta \bullet (P[\overrightarrow{\mathsf{R}} \bullet N/y])[\overrightarrow{\mathsf{S}} \bullet N/z] \text{ by IH}\end{aligned}$$

and the rhs:

$$\beta \bullet ((\delta^{y\,z}_x.P)^{\mathsf{a}_1}[N/x]) \to_{\lambda_{lcf}} \beta \bullet P[\overrightarrow{\mathsf{R}} \bullet N/y][\overrightarrow{\mathsf{S}} \bullet N/z]$$

where substitution associates to the left. □

Lemma 3.11 (local confluence) *There are seven criticalpairs in λ_{lcf} all of which are joinable.*

Proof. We distinguish the cases which are interesting w.r.t. the labels.

(i) Superposition of BETA-APP2: $((\lambda y.M)^\alpha N)^\beta[P/x]$, where $x \in \mathsf{fv}(N)$, gives rise to the critical pair $c_1 \approx c_2$ with

$$c_1 = \beta\overline{\overrightarrow{\mathsf{D}}\alpha\overleftarrow{!}} \bullet M[\underline{\overrightarrow{!}\alpha^r\overleftarrow{\mathsf{D}}} \bullet N/y][P/x] \qquad c_2 = ((\lambda y.M)^\alpha(N[P/x]))^\beta$$

which converges as shown below:

$$c_1 \to_{Cmp} \beta\overline{\overrightarrow{\mathsf{D}}\alpha\overleftarrow{!}} \bullet M[\underline{\overrightarrow{!}\alpha\overleftarrow{\mathsf{D}}} \bullet N[P/x]/y]$$
$$c_2 \to_{Beta} \beta\overline{\overrightarrow{\mathsf{D}}\alpha\overleftarrow{!}} \bullet M[\underline{\overrightarrow{!}\alpha\overleftarrow{\mathsf{D}}} \bullet (N[P/x])/y]$$

and $c_1 = c_2$ by the previous lemma.

(ii) For $y^\alpha[N/y][P/x]$, $c_1 = \alpha \bullet N[P/x]$ and $c_2 = y^\alpha[N[P/x]/y]$. After one application of the rule BETA we have $c_2 \to \alpha \bullet N[P/x] = c_1$

The remaining critical pairs arise from superpositions between APP1-CMP, APP2-CMP, CPY2-CMP, ERS2-CMP and CMP-CMP. Additionally, these critical pairs converge without requesting label sensitive rules (BETA, LAM, CPY1, ERS1, and VAR). □

Remark 3.12 The conditions on the rewrite rules are essential. If we drop the condition of the BETA rule then the overlap $((\lambda y.M)^\alpha N)^\beta[P/x]$, $x \in M$ gives rise to a critical pair which is not joinable:

$$\begin{tikzcd}
& ((\lambda y.M)^\alpha N)^\beta[P/x] \arrow[dddl] \arrow[dr] & \\
& & ((\lambda y.M)^\alpha[P/x]N)^\beta \arrow[d] \\
& & ((\lambda y.M[\overrightarrow{?} \bullet P/x])^\alpha N)^\beta \arrow[d] \\
\beta\overline{\overrightarrow{\mathsf{D}}\alpha\overleftarrow{!}} \bullet M[\underline{\overrightarrow{!}\alpha^r\overleftarrow{\mathsf{D}}} \bullet N/y][P/x] & \equiv^? & \beta\overline{\overrightarrow{\mathsf{D}}\alpha\overleftarrow{!}} \bullet M[\overrightarrow{?} \bullet P/x][\underline{\overrightarrow{!}\alpha^r\overleftarrow{\mathsf{D}}} \bullet N/y]
\end{tikzcd}$$

In this case, searching for a property of commutation of substitutions would not help because there is still a $\overrightarrow{?}$ in the branch on the right to deal with.

In order to show confluence, we will split λ_{lcf} into two relations: $\to_\sigma$ and $\rightrightarrows_{beta}$. Then we will make use of Rosen's lemma, which states that the union of two rewrite systems is confluent if both are confluent and commute.

- The first system consists of all σ-rules in λ_{lcf}. Notice that the critical pairs of $\to_\sigma$ are the ones presented earlier except the first one. Additionally, these critical pairs do not the request the BETA rule and therefore $\to_\sigma$ is locally confluent. Moreover, there are no infinite reduction sequences in $\to_\sigma$. Hence, $\to_\sigma$ is confluent.

- $\rightrightarrows_{beta}$ is defined to be the Beta rule of λ_{lcf} which contracts in parallel all beta redices at any position in a λ_{lcf}-term. To obtain confluence we show that $\rightrightarrows_{beta}$ has the diamond property (strong confluence), that is, if $M \rightrightarrows_{beta} M'$ and $M \rightrightarrows_{beta} M''$ then there exists N such that $M' \rightrightarrows_{beta} N$ and $M'' \rightrightarrows_{beta} N$. The proof is by induction on $M \rightrightarrows_{beta} M'$. Clearly, $\rightarrow_{beta} \subseteq \rightrightarrows_{beta}$ and $\rightarrow^*_{beta} = \rightrightarrows^*_{beta}$, which also makes the original beta rule confluent.

Proposition 3.13 (commutation of $\rightrightarrows / \sigma$) *If $M \rightrightarrows^*_{beta} M'$ and $M \rightarrow^*_{\sigma} M''$ then there exists N such that $M' \rightarrow^*_{\sigma} N$ and $M'' \rightrightarrows^*_{beta} N$.*

Proof. We first show that the diagram weakly commutes by induction on the definition of $\rightrightarrows_{beta}$. We distinguish the following cases:

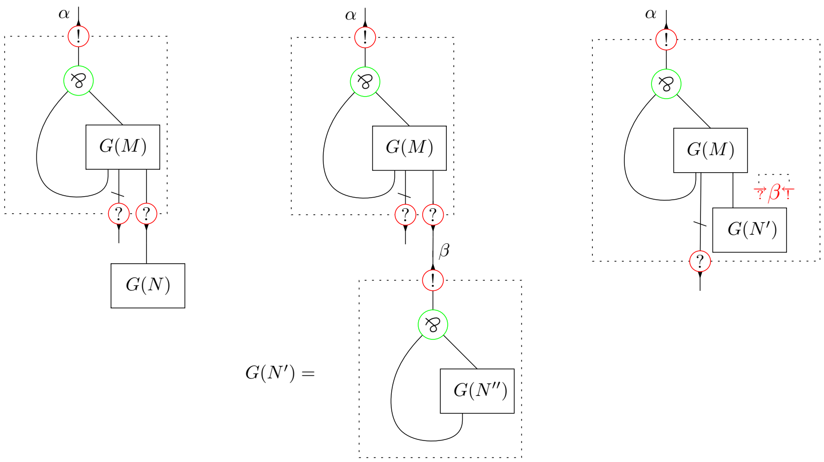
- Let $M = ((\lambda y.P)^{\alpha}O)^{\beta}$; $\rightrightarrows_{beta}$ applies at the root of the term, $M' = \beta\overline{\overrightarrow{\mathsf{D}}\alpha\overleftarrow{!}} \bullet P_b[\underline{\overrightarrow{!}\alpha^r\overleftarrow{\mathsf{D}}} \bullet O_b/x]$, $P \rightarrow_{beta} P_b$, $O \rightarrow_{beta} O_b$; $\rightarrow_{\sigma}$ is internal to P or O, $M'' = ((\lambda y.P_{\sigma})O)^{\beta}$ or $M'' = ((\lambda y.P)O_{\sigma})^{\beta}$ whenever $P \rightarrow_{\sigma} P_{\sigma}$ or $O \rightarrow_{\sigma} O_{\sigma}$. By IH , there exists a term P_v such that $P_b \rightarrow_{\sigma} P_v$ and $P_{\sigma} \rightarrow_{beta} P_v$ or there exists a term O_v such that $O_b \rightarrow_{\sigma} O_v$ and $O_{\sigma} \rightarrow_{beta} O_v$. Take $N = P_v[O_b/x]$ for the first case and $N = P_b[O_v/x]$ for the second case.
- Let $M = P[O/x]$, $M' = P_b[O_b/x]$, $M'' = P_{\sigma}[O/x]$ or $M'' = P[O_{\sigma}/x]$, where a σ-rule does not apply at the root of M and all reductions are internal to P or O. Hence the property holds by IH. On the other hand, if a σ-rule is applicable at the root of M then we have to consider different cases: a) we may have an overlap between a Beta rule (P is a beta redex) and the σ-rule. In this case only App2 can be applied since the functional part of the beta redex has to be closed. This case is similar to the first critical pair presented earlier where the pair converges at N. b) In any other case, take P to be the lhs (L) of one of the σ-rules, $M = L[O/x]$, $M' = L[O_b/x]$, $M'' = R[O/x]$; the diagram commutes with $N = R[O_b/x]$.

Commutation is obtained by induction on the length of the derivation $M \rightrightarrows^*_{Beta} M'$. Since $\rightarrow^*_{beta} = \rightrightarrows^*_{beta}$ we close the diagram and conclude that λ_{lcf} is confluent by Rosen's lemma. □

4 Structure of the labels

The markers in the labels add only *positions* of exponential information to the paths, that is, we still only know about the points of contact between multiplicative nodes (and consequently $D-!$ nodes). This is understood as an interaction between the functional part of the redex and the argument. The question we try to answer in this section is whether it is possible to mark the remaining exponential points of contact in the labels. The problem is that the remaining rules describe interactions between terms and substitutions. In particular, we would like to treat interactions in the rewrite rules Lam, Cpy1 and Ers1 as interactions with abstractions. We remind that all exponential cut rules involve a box structure which, in case of the lambda calculus translation, is an abstraction. A solution to our problem is to substitute only evaluated terms (i.e. simulate a call by value strategy) and ensure that substitutions evaluate to abstractions. It has been shown in [8] that closed terms in λ_{cf} evaluate to weak head normal form by simulating a call-by-value strategy. Notice that weak head normal forms of closed terms are also closed terms and thus the only possible results are abstractions. The conditions of the rules of interest require that substitutions are closed and this allows us to treat rules Cpy1, Lam and Ers1 as desired. Additionally, we remind that the rule Beta also involves a closed dereliction cut and this completes the picture of exponential cut elimination steps that we can simulate via our reasoning.

Example 4.1 In the following example we show how to apply these ideas to the rule Lam: we first evaluate the substitution of the left hand side of the rewrite rule and then move to the right hand side of the rule. This simulates a closed commutative cut elimination step and we can capture this situation in the label as follows:



We usually add labels to the paths once nodes have been removed from the graph. For this example only, we slightly abuse this intuition and add a trailing Of Course label to mark the points of contact. Notice that the root of the box in $G(N')$ has survived the reduction and is subject to further exponential cut elimination steps until the box gets involved with a dereliction cut.

However, it is not necessary to simulate a call-by-value strategy to identify these points of contact. It turns out that the structure of the multiplicative overlining and underlining provides enough information to recover the exponential connectivity. As in Lévy's calculus, overlining and underlining does not only add connectivity to the labels but also depth.

Definition 4.2 We define *depth*(ϕ) to be the overall nesting of a label. This is the total number of overlines and underlines that surround a label.

By adding depth to a label, we achieve to place the sub-label that is irrelevant to further Beta reductions into history.

Lemma 4.3 *Let $L(M)$ denote the external label of a term. By using only the rules* *in σ, if $M \to_\sigma^* M'$ then $depth(L(M)) = depth(L(M'))$*

Proof. Substitutions add only *positions* for Why Not, Contraction and and Weakening nodes. It is the BETA rule that adds depth to the labels. □

Additionally, due to confluence of σ, the positions in which these markers are recorded do not depend on the order in which we apply the substitutions.

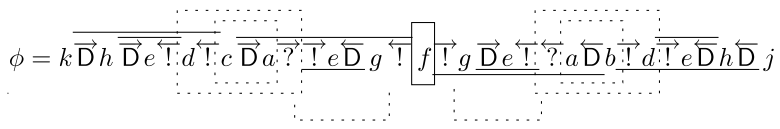
Proposition 4.4 (Virtual Cuts) *Let γ be an overlined (resp. underlined) sublabel at depth k and let Exp be the multiset of all atomic exponential markers at depth $k+1$. The multiplicity of $\overrightarrow{!} \in Exp$ (resp. $\overleftarrow{!} \in Exp$) is 1. Each exponential marker other than the Of Course marker in the multiset forms a virtual exponential cut with that Of Course marker.*

Proof (Sketch) From the previous discussion, it follows that only labels with depth 0 play a role in further BETA reductions. We consider the situation where we apply σ-rules only in which case exponential markers ?, W and R (resp. S) are added at depth 0 and retain this depth by the previous lemma. The ! marker that forms the other end of the cut has to be located in a relevant portion of the label, that is, at depth 0. But this marker is provided only during a BETA rewrite step, which also places a D marker at depth 0 (marking the final dereliction cut that destroys the box) and then increments all depths by surrounding the label with an overline / underline. □

Example 4.5 Here we repeat the example of the labelled reduction of $(\lambda xy.xy)II$ where we now add exponential connectivity to the paths.

$$(((\lambda x.(\lambda y.(x^a y^b)^c)^d)^e(\lambda x.x^f)^g)^h(\lambda x.x^i)^j)^k \to^*_{\lambda_{lcf}} (\lambda x.x^i)^\phi$$

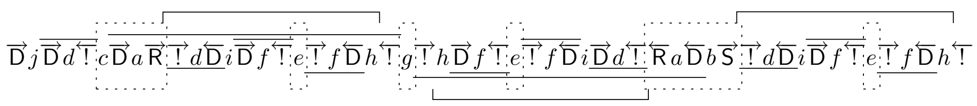
with



where we highlight the boundaries of the box and the virtual commutative (!−?) cut. Notice that the direction of the arrow of each Of Course marker faces the direction of the arrow of the exponential marker with which it forms a cut. A slightly more complicated example is the reduction of the term $(\lambda x.xx)(II)I$, which shows how information flows through contraction nodes:

$$(((\lambda x.(\delta_x^{p,o}(p^a o^b)^c))^d((\lambda x.x^e)^f(\lambda x.x^g)^h)^i)^j(\lambda x.x^k)^l)^m \to^*_{\lambda_{lcf}} (\lambda x.x^k)^\phi$$

with $\phi = m\overline{\psi}g\underline{\psi}^r l$ and ψ is given below:



We have split the contraction nodes into R and S in order to capture the different routes that paths can take through contraction nodes. The corresponding graph is presented in Figure 2.

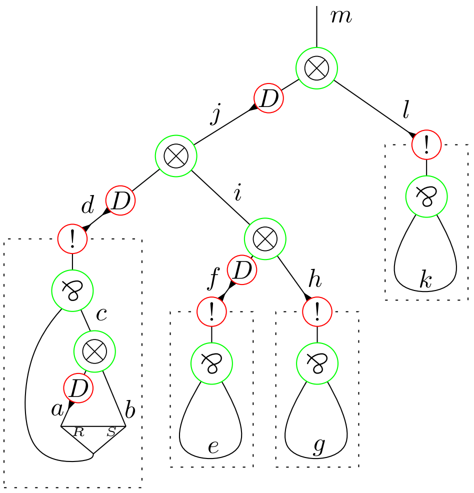


Fig. 2. Translation of $(\lambda x.xx)(II)I$

5 Towards Closed Reduction in the GOIm

The main interest in examining strategies for the GOIm comes from the fact that the length of the paths to be traversed can be significantly reduced. The reader should notice that strategy here means computing the same paths differently. For instance, the Jumping Abstract machine presented in [6] shortcuts paths that are underlined: these are the same as the overlined paths, but in reverse order. The Call-By-Value GOIm presented in [7] adds on top of this idea and shortcuts sub-redundancies like the exponential virtual cuts, which are highlighted in our second example. These sub-redundancies describe repeated evaluations of arguments. Notice that this level of redundancy is not immediately present in Lévy's labels.

A common approach that has been adopted to achieve these optimisations is to borrow features from environment machines. The Jumping Abstract machine and the Call-By-Value GOIm have been related to Krivine's (call-by-name) machine and to the Categorical Abstract Machine (CAM) respectively. The approach is to take "snapshots" of the environment during the traversal of the graph such that this environment is restored at the time a shortcut is taken. The price to pay is that copying and discarding environments is an expensive operation. On the other hand, the Geometry of Interaction machine is free from these deficiencies but does not take any shortcuts.

Our interest in focusing on a closed strategy is in that the dynamics of closed cut elimination are much simpler than in the full case but powerfull enough for implementing functional languages. An important point to notice is that the Call-By-Value GOIm demonstrates that one has not to follow the paths exactly in the same order as they appear in the labels (from left to right).

Here we provide an informal description on how a closed strategy acts. Computation is initialised by following a normal flow in the traversal of the graph but once the strategy detects the appearance of ?-nodes in boxes, the normal flow is put on hold in order to deal with the ?-nodes first. The goal is to remove ?-nodes from the graph by installing shortcuts such that ?-nodes are practically invisible once the normal flow of the computation resumes. In terms of Linear Logic, the strategy tries to close boxes as soon as possible in order to allow only closed cut elimination steps to take place.

We recall the graph and the label from our first example (4.5) for a more concrete situation: Computation starts by traversing the edges $kDhDe!d$. Up to this point, we follow exactly what the label ϕ dictates. Since we arrive at a box that contains a ?-node we have to initialise a search starting from that ?-node to find the !-node with which it forms a virtual commutative cut. The path that we traverse is $?!eDg$ which leads us to the matching !-node. Now we rewire the edge a of the ?-node with the root of the matching !-node and resume the normal flow of the computation. This was at edge d. The path that we traverse from now and on is exactly what the label ϕ dictates but now we jump all virtual commutative cuts that are highlighted in the label.

6 Conclusion

In this work we presented a calculus whose labels provide sufficient structure in order to experiment with new strategies for the Geometry Of Interaction machine. The calculus unveils a fine grained level of redundancy in the labels and also provides us with a tool that will help to prove properties of a closed GOI machine.

It is important to note that our calculus can only capture exponential information for the call-by-value translation. If we try to use the call-by-name translation of Linear Logic instead, then our calculus would run into problems. This is because the call-by-name translation places the exponential box structure in argument positions. As pointed out earlier, the multiplicative information in the labels is not affected by this change but the exponential markers will have to reflect the new situation. An extension of the labels to the call-by-name translation and the comparison of the structures obtained from both approaches is left for future work.

7 Acknowledgements

We thank Ian Mackie for his contributions throughout this work and our anonymous referees for their helpful comments.

References

[1] A. Asperti, V. Danos, C. Laneve, and L. Regnier. Paths in the lambda-calculus. In *Logic in Computer Science*, pages 426 – 436, 1994.

[2] A. Asperti and S. Guerrini. *The optimal implementation of functional programming languages*. Cambridge University Press, New York, NY, USA, 1998.

[3] A. Asperti and C. Laneve. Interaction systems i: The theory of optimal reductions. *Mathematical Structures in Computer Science*, 4(4):457–504, 1994.

[4] A. Asperti and C. Laneve. Paths, computations and labels in the lambda-calculus. In *RTA-93: Selected papers of the fifth international conference on Rewriting techniques and applications*, pages 277–297, Amsterdam, The Netherlands, The Netherlands, 1995. Elsevier Science Publishers B. V.

[5] T. Blanc, J.-J. Lévy, and L. Maranget. Sharing in the weak lambda-calculus. In A. Middeldorp, V. van Oostrom, F. van Raamsdonk, and R. C. de Vrijer, editors, *Processes, Terms and Cycles*, volume 3838 of *Lecture Notes in Computer Science*, pages 70–87. Springer, 2005.

[6] V. Danos and L. Regnier. Reversible, irreversible and optimal lambda-machines. *Theoretical Computer Science*, 227:79 – 97, 1999.

[7] M. Fernández and I. Mackie. Call-by-value lambda-graph rewriting without rewriting. In *ICGT '02: Proceedings of the First International Conference on Graph Transformation*, pages 75–89, London, UK, 2002. Springer-Verlag.

[8] M. Fernández, I. Mackie, and F.-R. Sinot. Closed reduction: explicit substitutions without α-conversion. *Mathematical. Structures in Comp. Sci.*, 15(2):343–381, 2005.

[9] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[10] J.-Y. Girard. Geometry of interaction 1: Interpretation of System F. In R. Ferro, C. Bonotto, S. Valentini, and A. Zanardo, editors, *Logic Colloquium 88*, volume 127 of *Studies in Logic and the Foundations of Mathematics*, pages 221–260. North Holland Publishing Company, Amsterdam, 1989.

[11] J.-J. Levy. Reductions correctes et optimales dans le lambda-calcul. these de doctorat d'etat, universite paris vii. 1978.

[12] I. Mackie. The geometry of interaction machine. In *POPL '95: Proceedings of the 22nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 198–208, New York, NY, USA, 1995. ACM Press.

[13] I. Mackie. Interaction nets for linear logic. *Theor. Comput. Sci.*, 247(1-2):83–140, 2000.