URL: http://www.elsevier.nl/locate/entcs/volume3.html 21 pages

Reversible, Irreversible and Optimal λ-machines

Extended abstract

Vincent Danos

Equipe de logique de Paris 7
CNRS, Université Paris 7
Paris, France

Laurent Regnier

Institut de Mathématiques de Luminy
CNRS
Marseille, France

Abstract

There are two quite different possibilities for implementing *linear head reduction* in λ-calculus. Two ways which we are going to explain briefly here in the introduction and in details in the body of the paper. The paper itself is concerned with showing an unexpectedly simple relation between these two ways, which we term *reversible* and *irreversible*, namely that the latter may be obtained as a natural optimization of the former.

Keywords: λ-calculus, abstract machines, geometry of interaction, reversible computations.

1 Introduction

Notation.

We denote the application of U to V by $(U)V$, *e.g.*, the Church integer $\bar{2}$ will be $\lambda f \lambda x\,(f)(f)x$.

Linear head reduction.

But what is exactly linear head reduction, to begin with. It is a variant of head reduction, where one substitutes at each step the leftmost *occurrence* of variable whenever it is engaged into a redex, as in:

$$\begin{aligned}(\lambda f\,(\underline{f})(f)x)\lambda y\,y &\to (\lambda f(\lambda y\,\underline{y})(f)x)\lambda y\,y\\ &\to (\lambda f(\lambda y\,(\underline{f})x)(f)x)\lambda y\,y\\ &\to (\lambda f(\lambda y\,(\lambda y\,\underline{y})x)(f)x)\lambda y\,y\\ &\to (\lambda f(\lambda y\,(\lambda y\,\underline{x})x)(f)x)\lambda y\,y\end{aligned}$$

where the successive leftmost occurrences of variables are underlined. Note that terms always grow by this reduction. This hyper-lazy reduction, which does the minimal effort to try and get a (quasi-) head normal form is, nevertheless, easily seen to be strong enough to evaluate terms in data types provided one considers terms up to the equivalence generated by $(\lambda x\lambda y\,T)U = \lambda y(\lambda x\,T)U$ when $y \notin U$. This equivalence is half of the σ-equivalence defined in [15]. It is shown there that two σ-equivalent terms have the same length of head reduction, leftmost reduction and longest reduction. So σ-equivalence, and *a fortiori* the finer equivalence we use here, is really a mild quotient on terms.

Note that if a term is normal with respect to linear head reduction, up to the equivalence above, it is in head normal form, except for some head redexes not concerning the leftmost variable which may remain, waiting to be triggered to get an actual head normal form. In the example:

$$\begin{aligned}(\lambda f(\lambda y\,(\lambda y\,\underline{x})x)(f)x)\lambda y\,y &\to (\lambda y\,(\lambda y\,\underline{x})x)(f)x\\ &\to (\lambda y\,\underline{x})x\\ &\to \underline{x}\end{aligned}$$

An irreversible linear head reduction machine.

The KAM, or *Krivine's abstract machine*, is by far the simplest evaluation mechanism for λ-calculus. Its state is a triple, U, E, S where: 1) the term U is a subterm of the global term under evaluation; 2) the stack S contains *closures* (that is pairs (U, E) consisting in a subterm and an environment) that still have not found the variable they match or "fall into"; 3) and the environment is a list of $(x \to U, E)$ giving values for variables free in U (and for others as well but it won't use them any more). The KAM repeatedly goes for the leftmost variable of the current subterm, storing information along its way:

$$\frac{(T)U:\quad E\quad S}{T:\ E\quad (U,E).S}\qquad\qquad \frac{\lambda x\,T:\quad E\quad (U,E').S}{T:\ (x\to U,E').E\quad S}$$

so that when hitting that leftmost variable it can retrieve a subterm, together with an appropriate environment, where to start again the process:

$$\frac{x:\ \ldots(x\to U,E)\ldots\quad S}{U:\qquad E\qquad S}$$

To make this definition precise, one has to say that in the value retrieval step, or jump step, the value to be fetched in the environment sits at the n-th place where n is x's de Bruijn index (the number of binders in the scope of which x is free). Because there may be many values at hand in an environment for the same occurrence of variable.

Likewise one can define an elegant environment machine, the PAM, or *pointer abstract machine*, which builds no closures, but relies during the value retrieval step on x's Böhm index (the number of applications in the right part of which x is free) rather than on its de Bruijn index, to fetch a value.

When it turns to implementing the KAM, it would be foolish to actually duplicate the environment in the application step, it is just a pointer here which gets duplicated. But then, in the jump step, one can't free the memory space allocated to the part of the environment which the jump discards, and consequently, the whole process is inflationary, that is the amount of information collected grows steadily at each step. This is what we mean when saying that the mechanism is *irreversible.* It has to call for an external garbage-collection mechanism to dispose of the obsolete information. Just the same happens for the PAM.

A reversible linear head reduction machine.

The IAM, or *interaction abstract machine*, comes from Girard's *geometry of interaction* interpretation of terms as partial isometries, or, more to the earth, as partial one-one transformations on a countable base-set. It turns a term into a sort of bi-deterministic, or reversible, finite state automaton, but with somewhat more general transition steps since more one-one partial transformations on words are allowed, than the traditional mere popping of the first letter. A *run* then consists in entering the automaton with a word at some given entry node and then traveling inwards modifying that word according to the transformations encountered.

As the automaton is *bi-deterministic*, that is: given a word and a node, no two transformations apply, nor could two transformations have pushed the word there, the amount of information needed to keep the machine running, which is but the word, will sometimes grow and sometimes shrink. We say, by contrast, that the mechanism is *reversible.* It seems that it needs no external garbage-collection.

Contents of the paper.

We shall uncover in this paper a simple and hidden relation between these two mechanisms. First, we will proceed to the definition of the IAM in the somewhat more general frame of Linear Logic *nets.* We will next lead an analysis of its runs, based on Asperti and Laneve's work on Lévy's labelings and the structure of the paths these labelings denote, and conclude to the existence of a highly natural optimization of the IAM: the JAM, or *jumping abstract machine*, which shortcuts redundant steps.

Only then will this new machine be specialized to the particular case of λ-calculus, according to the two basic embeddings of λ-calculus in nets. If one uses the equation $D = !D \multimap !D$ then the specialized JAM happens to be the KAM, and if one uses $D = !D \multimap D$ then it happens to be the PAM. Thus the KAM and the PAM are seen to be two instances of the same machine, but using a different subset of types.

Where ideas came from.

Part of the results, namely that the PAM jumps along the path that the token follows in the IAM, was known to us since 89; also it was independently discovered by Malacaria in 91, but for the KAM (private communication). But the missing half, namely that reversible computations could be taught to jump from the inside, had to wait until the discovery of the call/return symmetry due to Asperti and Laneve in 92. The last hint for solving that riddle was to use a variant of the diode/context semantics (which Mackie also uses in [13] to simplify compilation) that makes it clear how to shortcut the returns (see lemmas 4.1 and 4.5).

Expectations.

The reader will also want to know whether this may lead to something really new. Well it did already. We recognized in [4] that the interaction processes at work in Hyland and Ong (HO) and Abramsky, Jagadeesan and Malacaria (AJM) respective new game semantics were precisely the PAM and the IAM. The link here disclosed between the two machines helped in the construction of an embedding of AJM-games into HO-games, which in turns gives a simple proof of definability, or full abstraction, for AJM strategies.

But maybe the more interesting still lies ahead. From the JAM, as was suggested by Asperti, it is quite easy to devise a mechanism which will allow jumps not only for return paths, but will add on-the-fly edges embodying paths corresponding to redex families as soon as they are detected (*i.e.*, completed for the first time). Could it be an answer to the search for efficient optimal implementations ?

2 Preliminaries: nets, paths and duality

Nets.

We only give here a brief description of nets, for more details on correctness conditions, elementary steps of reduction (the analog of β-reduction for nets) and properties of reduction, see [7,5,3,14,12].

Nets are oriented graphs built over an alphabet of nodes, or *links*, with edges typed by formulae of linear logic. Each link has a given number of incident edges called the *premises* of the link and a given number of emergent edges called the *conclusions* of the link.

axiom: no premise and two conclusions typed by dual formulae;

cut: two premises typed by dual formulae and no conclusion;

par and tensor: the *multiplicative* links have two premises and one conclusion typed by the **par** or the **tensor** of the premises;

of course: one premise and one conclusion typed by the **of course** of the formula typing the premise;

dereliction: one premise and one conclusion typed by the **why not** of the formula typing the premise;

weakening: no premise and one conclusion typed by a **why not** formula;

contraction: two premises and one conclusion typed by the same **why not** formula;

pax: one premise and one conclusion typed by the same **why not** formula.

Edges which are not premise of a link are the *conclusions* of the net.

Conditions on nets.

Nets are required to fulfill two additional conditions:

Box condition: to each **of course** link $\mathtt{n}$ in the net R is associated a subnet b of R, called a *box*, such that one conclusion of b is the premise of $\mathtt{n}$. We call $\mathtt{n}$ the *principal door* of b. All the other conclusions of b are premise of **pax** links $\mathtt{a}_i$ in R. We call the $\mathtt{a}_i$'s the *auxiliary doors* of b. Each **pax** link in R must be auxiliary door of exactly one box. Two boxes are either disjoint or included one in the other.

Sequentialization condition: any net may be built by induction; that is, a net is either an **axiom**, or the **tensor** or **cut** of two nets, or the **par**, **dereliction**, **contraction**, **weakening** or boxing of one net. Purely geometrical conditions known as *correctness conditions* exist that are equivalent to the inductive one.

The reader will check that the nets built by the translation of λ-calculus in section 6 satisfy those two conditions. The *depth* of a node is the number of boxes it belongs to.

Paths.

If e is an edge we denote by $e^\star$ its *reverted* edge, *i.e.*, the edge oriented from the goal of e to the source of e.

A path is any sequence of edges and/or reverted edges such that, as usual, the goal of any edge is the source of its successor in the sequence, if any. If φ is a path we denote by $\varphi^\star$ its reverse.

A *straight* path must verify the additional condition that direction switchings only happen in **cut** and **axiom** nodes, *i.e.*, whenever $e_1 e_2^\star$ (resp $e_1^\star e_2$) belongs to the path, then e_1 and e_2 are the two conclusions (resp. premises) of a **cut** (resp. **axiom**) node. All paths are now supposed to be straight ones.

Let `n` be a node, which is neither a **cut** nor an **axiom**, and let φ be a path. If φ contains an edge (resp. a reverted edge) adjacent to `n` (*i.e.*, such that the source or the goal is `n`) then we say that φ crosses `n` *downwardly* (resp. *upwardly*).

Exponential tree, branches and lifts.

An exponential tree is a maximal subtree of a net with edges typed by the same **why not** formula. Thus the leaves of an exponential tree are **weakening**, **dereliction** and **axiom** links; the inside nodes of an exponential tree are **contraction** and **pax** links. An *exponential path* is a path starting from any node of an exponential tree, and moving downward to its root. An exponential *branch* γ is an exponential path starting from a **dereliction** link; γ's *lift* is the number of **pax** links that γ crosses.

Equations on formulae preserving duality.

Possibly, *e.g.*, when encoding pure λ-calculus in linear logic (see section 6 for such an encoding), one needs to quotient formulae by an appropriate equivalence relation. So doing, one has to prove that the equivalence preserves duality at the level of normalization. That is to say, termination may well be lost, for instance if using $O = !O \multimap O$ or $O = !O \multimap !O$ which both allows for a faithful encoding of pure λ-calculus, *but*, local cut eliminability is preserved. The equations just mentioned do preserve duality in our specialized sense, and so does in general $O = F[O]$ for any linear logic formula F.

3 The interaction abstract machine

Stacks.

Let a *stack* be any finite sequence of:

(i) *multiplicative constants*, P and Q;

(ii) *exponential signatures*, which are binary trees with leaves labeled by the *exponential constants* P', Q' and $\Box$.

Notations.

We denote by $:$ and ε the stack constructors and by $\cdot$ the binary tree constructor; we will use σ, σ' to range over signatures; Σ will be the set of stacks equipped with Cantor's topology.

Actions.

Let $\mathcal{A}$ be the set of *actions*, that is partial and continuous one-one transformations on $\Sigma \times \Sigma$. With the composition and the inversion the set $\mathcal{A}$ of actions has the structure of *inverse monoid*, *i.e.*, satisfies:

$$\begin{aligned} (x^*)^* &= x \\ xx^*x &= x \\ xx^*yy^* &= yy^*xx^* \end{aligned}$$

for any actions x and y.

We will use (B, S) to range over pairs of stacks on which actions act; B, will be termed the *boxes stack*, and S, the *balancing stack*.

Concretely, such actions (because they are continuous) can be finitely presented as finite sets of clauses with non unifiable heads (because they are maps) and non unifiable bodies (because they are one-one). Composition, with this representation, is resolution, and inversion is simply exchanging head and body. See the "resolution algebra" in [11].

Attaching actions to edges.

To each (oriented) edge e of a net, one associates an action $a(e)$ in $\mathcal{A}$ depending on the link of which e is a premise:

cut: if e is premise of a cut then $a(e)$ is the identity on (B, S);

multiplicative: if e is the left (resp. right) premise of a multiplicative link then $a(e)$ is p (resp. q) defined by:

$$p(B, S) = (B, P : S) \qquad q(B, S) = (B, Q : S)$$

dereliction: if e is the premise of a dereliction link then $a(e)$ is d defined by:

$$d(B, S) = (B, \Box : S);$$

contraction: if e is the left (resp. right) premise of a contraction link then $a(e)$ is p' (resp. q') defined by:

$$p'(B, \sigma : S) = (B, (P' \cdot \sigma) : S) \qquad q'(B, \sigma : S) = (B, (Q' \cdot \sigma) : S)$$

pal: if e is the principal door of a box then $a(e)$ is b defined by:

$$b(\sigma : B, S) = (B, \sigma : S);$$

pax: if e is the auxiliary door of a box then $a(e)$ is the sequence of actions tb where b is already defined and t is given by:

$$t(B, \sigma : \sigma' : S) = (B, (\sigma \cdot \sigma') : S).$$

Format of stacks.

Note that b is the only action above which acts on the stack B, everything else happening on S. Hence if the height of B is once the depth of the node where (B, S) stands, it was and will be so. Whence the name "boxes stack" for B. We ask all boxes stacks to satisfy this constraint called that *depth invariant.*

We may note also, in the typed case, that is when no quotient is performed on formulae, a similar constraint on the height of the balance stack. Let S^e (resp. S^m) denote S minus all multiplicative constants (resp. exponential signatures). Given A a formula, to each atom X of A, one associates one-onely an S^m by induction on A: if $A = X$ then $S^m = \varepsilon$, if $A = \Box A_1$, where $\Box$ is an exponential, then $S^m_A(X) = S^m_{A_1}(X)$, and if $A = A_1 \bullet A_2$ and $X \in A_1$ (resp. $X \in A_2$), then $S^m_A(X) = P : S^m_{A_1}(X)$ (resp. $Q : S^m_{A_2}(X)$). Now, if once, S^m denotes an atom X in the type of the current node, say A, and if the height of S^e is the number of exponentials in A in the scope of which X stands, then it was and will be so.

By the way multiplicative constants can be treated separately with a specialized third stack, but it wouldn't be as convenient technically.

Attaching actions to edges can be seen as equipping the net with a structure of extended automaton. The rôle of words is played here by pairs of stacks, that of transitions by actions; reduction then may be seen as a minimization process. It is easy to see that the attachment here chosen turns the net in a bi-deterministic automaton. Thinking it over, it becomes even clear, that this attachment is the *one* natural bi-deterministic automaton structure with which one can equip links, so that both constraints on heights, as explained above, are satisfied. Here, we won't use the S-constraint, which is just mentioned in order to make obvious the canonicity of this attachment, which the reader could otherwise find quite arbitrary.

Axioms for such nets with a reversible extended automaton structure on the top of it were given in [6] together with the definition of a local reduction on them, in order to study optimal machines.

Action of a path.

The mapping $a(.)$ extends to a functor from the free $*$-category of paths in the net R into the inverse monoid $\mathcal{A}$. More explicitly, if e is a an oriented edge one sets $a(e^*)$ to be $a(e)^*$, and then to any path ϕ one associates the action $a(\phi)$ obtained by composing the actions associated to each (reversed or not) oriented edge crossed by ϕ. Note that $a(\phi)$ may be nowhere defined, *e.g.*, if ϕ is $e_1 e_2^*$ where e_1 and e_2 are the two premises of a multiplicative link then $a(\phi) = q^* p$ is the empty map. If this is not the case, *i.e.*, if for some (B, S), $a(\phi)(B, S)$ is defined, then one says ϕ is *regular.*

The interaction abstract machine.

Let R be a net.

A *run* of the IAM on R consists in an *initial pair* (ε, S) and a path ϕ starting upward in a conclusion of R such that $a(\phi)(\varepsilon, S)$ is defined. A run is *successful* if ϕ ends downward in a conclusion. By determinicity, an initial pair defines at most one successful run. Observe also that ϕ in a run is always regular, by definition.

Let $ex(R) = (e_{ij})$ stand for the matrix indexed by the conclusions of R with $e_{ij} \in \mathcal{A}$, such that a coefficient $e_{ij}(\varepsilon, S) = (\varepsilon, S')$ iff there is a successful run starting in the i-th conclusion with the initial pair (ε, S) and ending in the j-th conclusion with the final pair (ε, S'). This is Girard's execution formula rephrased as appropriate in our framework; see [8–10] for the original presentation.

This may seem a formidable thing to compute, but remember that all actions are finitely representable, and then it is easy to come up with a finitary formulation of this $ex(R)$. Note also that by bi-determinicity $ex(R)$ is a self-dual action matrix.

Now, take note, $ex(R)$ is *not* an invariant of net reduction, but it is an invariant of *closed* reduction, that is reductions of nets where exponential steps handle boxes with no auxiliary doors.

Output of a run.

Up to this point, it may be hard to see in which sense this IAM is able to compute something. To get some output one needs to introduce data-types, and new links to represent constants and functions. These new nodes will be of the same shape as axioms, in that they will have no premise.

Without functions, it is easy to show that any net of ground type, say o, will have a unique successful run ϕ starting with the empty initial pair $(\varepsilon, \varepsilon)$ at the conclusion of type o and ending in the constant node which is the actual value of the net. Because a such net must have a closed reduction to its value. And then the fact that ϕ ends at the right value node follows from the invariance of $ex(R)$ under such closed reductions. With functions, one has to add a side-effect stack where values are handled and given to functions. This strategy of "computing by paths" is extended to PCF, a language with in addition, conditionals and fixpoints, in [13].

4 Well balanced paths and !-cycles

In this section, we will set the stage for an optimization of the IAM defined in the previous section. This optimization relies on the fact that runs in general enter in redundant steps, which we are first going to identify and then to shortcut.

In the sequel we will sometimes simply write $\phi(B, S)$ for $a(\phi)(B, S)$.

4.1 Well balanced paths.

Let say two edges are dual when they are typed by dual formulae. A *well balanced path*, wbp for short, is a path starting downward with an edge e and ending upward with a dual edge $f^\star$, and inductively given by:

Cut: if e and f are premises of a same **cut** link, then $ef^\star$ is a wbp;

Reversion: if φ is a wbp, then its inverse $\varphi^\star$ is a wbp;

Multiplicative stretching: if φ is a wbp connecting a **tensor** link to a **par** link, e is the left (resp. right) premise of the **tensor** and f is the left (resp. right) premise of the **par** then $e\varphi f^\star$ is a wbp;

Exponential stretching: if φ is a wbp connecting the root of an exponential tree t to an **of course** link and γ is an exponential branch of t starting from a **dereliction** link $\mathtt{d}$ the premise of which is e and f is the premise of the **of course** link then $e\gamma\varphi f^\star$ is a wbp;

Composition: if φ_1 is a wbp ending in a conclusion of an **axiom** link $\mathtt{a}$ and φ_2 is a wbp starting with the other conclusion of $\mathtt{a}$ then $\varphi_1\varphi_2$ is a wbp.

This definition is equivalent to [2]'s one, yet is simpler because the latter mixes stretching and composition. Note that composition is the only clause of the definition that may generate unregular wbp's.

Lemma 4.1 (Balance property) *Let φ be a wbp; then there is a partial and continuous one-one transformation $\tilde{\varphi}$ such that, for any B and S:*

$$a(\varphi)(B, S) = (\tilde{\varphi}(B), S).$$

The lemma is easily checked by induction on the definition of wbp's. Another easy induction on paths, yields:

Lemma 4.2 *Let ϕ be a path, and B a stack, such that $\phi(B, S)$ is defined for all S, then there are constants and signatures $x_1, \ldots, x_n$, and a stack B', such that $\phi(B, S) = (B', x_1 : \ldots : x_n : S)$.*

Lemma 4.3 (Converse Balance property) *Let φ be a path that ends upward and begin downward. If:*

$$\exists B, B', \forall S : a(\varphi)(B, S) = (B', S) \tag{1}$$

then φ is a wbp.

Proof. One argues by induction on the length of φ. Note that the hypotheses φ begins downward, ends upward and (1) are true for φ iff they are true for $\varphi^\star$.

1. Suppose there is a proper prefix φ_0 of φ that already satisfies the hypothesis, and suppose also there is an edge f such that $\varphi = \varphi_0 f^\star \ldots$, then the target node of f can't be a multiplicative or exponential node, because $f^\star$ in such cases is never defined for all S's, and we know φ is. It cannot be a cut node either, since a cut node cannot be reached upwardly. Consequently, one must have $\varphi = \varphi_0 f \ldots$, and f must be the conclusion of an axiom node, since only there a path may change its orientation. But then $\varphi = \varphi_0\varphi_1$ with both paths satisfying the hypothesis, and so by induction, φ is a wbp obtained by the composition clause.

2. Suppose now there is no such proper prefix, and put $\varphi = e\varphi' f^\star$.

2.a. If e is premise of a cut node, then φ' is empty and $\varphi = ef^\star$ is produced by the first cut clause; because, else, the first two edges of φ would form a proper prefix satisfying the hypothesis.

2.b. If e is not, then φ' can't be empty and must begin downward. Dually, φ' must also end upward, else $f^\star$ would be premise of a cut node, and hence form a proper suffix satisfying the hypothesis with the last edge of φ'.

Now observe, that e and f must be either **of course**, **dereliction** or multiplicative premises, since only those are defined for all S's. In case e (resp. f) is a **dereliction** premise, with associated exponential branch γ_e (resp. γ_f), then set $e' = e\gamma_e$ (resp. $f' = f\gamma_f$), else $e' = e$ (resp. $f' = f$). One gets φ decomposed as:

$$(B,S) \overset{e'}{\to} (B_1, x:S) \overset{\varphi'}{\to} (B'_1, y:S) \overset{f'^\star}{\to} (B', S)$$

for some constants or signatures x and y. Put φ'' to be the longest prefix of $\varphi' f^\star$ such that for all S_1, $\varphi''(B_1, S_1)$ is defined. By the lemma above, one knows there are $x_1, \ldots, x_n$, and a stack B_2, such that $\varphi''(B_1,S_1) = (B_2, x_1 : \ldots : x_n : S_1)$. If $\varphi'' = \varphi' f^\star$, then $\varphi(B,S) = (B_2, x_1 : \ldots : x_n : x : S)$, which is absurd, so φ'' is a prefix of φ' in fact. By maximality, one has that $n = 0$ else it is always possible to extend φ'', and we just said that φ'' couldn't be extended beyond the end of φ'. Again by maximality, there is an **of course** premise, or an exponential branch, or a multiplicative premise, say g, such that $\varphi = e'\varphi'' g^\star \ldots$; whence, $e'\varphi'' g^\star(B,S) = (B_3, S)$, and since φ has no proper such suffix, it must be that $\varphi'' = \varphi'$. But then φ' is a wbp, by induction, and the situation is simplified in:

$$(B,S) \overset{e'}{\to} (B_1, x:S) \overset{\varphi'}{\to} (B'_1, x:S) \overset{f'^\star}{\to} (B', S)$$

It remains to prove that φ is obtained from φ' by stretching: if $e = e'$ is a multiplicative premise, then $f = f'$ must be of the same form, and likewise if $e' = e\gamma_e$, then $f = f'$ must be an **of course** premise by duality (even in the untyped case, see the discussion at the end of the preliminaries). □

Lemma 4.4 (Sub-wbp property) *Let φ be a wbp and* n *a link crossed upwards by φ, then there is a unique maximal sub-wbp φ_0 of φ ending upward in* n.

Proof by induction on the definition of wbp's.

4.2 !-strings and !-cycles.

Let b, b' be boxes in a net; a *b-string* is a path ψ starting upward and ending downward, inductively defined by:

Basic b-string: a basic b-string is a path starting upward and ending downward and entirely contained in b;

General b-string: let ψ_0, ψ_1 be b-strings, γ be an exponential branch exiting b, φ be a wbp, and ψ' be a b'-string, then $\psi_1\gamma\varphi\psi'\varphi^\star\gamma^\star\psi_0$ is a b-string.

When a b-string ψ starts and ends in the principal door of b we call it a *b-cycle* (and not a bicycle), or simply a !-cycle.

Lemma 4.5 (Boxes property) *Let ψ be a !-cycle; then there are a partial one-one transformation $\tilde{\psi}$ and a partial identity π such that for any stacks B, S and any exponential signature σ:*

$$a(\psi)(B,\sigma\!:\!S) = (\pi(B),\sigma\!:\!\tilde{\psi}(S))$$

As the balance property, the boxes property is easily proved by induction on the definition of !-cycles. Those two lemmas are the IAM versions of the rendez-vous and the !-cycle properties in [2].

Lemma 4.6 (!-cycle suffix property) *Let R be a net, let ϕ be a path in R ending downward in the principal door of a box b, and s be a state such that $\phi(s) = (B',\sigma : S')$ for some σ created along ϕ, then there is a !-cycle ψ which is a suffix of ϕ.*

Proof. To prove this, let us define ψ_0 to be the suffix of ϕ starting when ϕ entered b for the last time. If this was by the principal door, then $\psi = \psi_0$ and we are done. Else, let γ be the exponential branch which ϕ uses to enter b at that time, and put $(B,\gamma(\sigma_1,\ldots,\sigma_p) : S_0)$ to be the state just before climbing up γ. Note that σ must be one of the σ_i's, otherwise by the depth invariant we see that the final state of ϕ cannot be $(S',\sigma : B')$. Put φ to be the longest path such that $\varphi^\star\gamma^\star\psi_0$ is a suffix of ϕ on the one hand, and on the other hand $\varphi(B,S)$ is defined for *all* S.

So that, by lemma 4.2, $\varphi(B,S) = (B',x_1 : \ldots : x_n : S)$ for all S. Since σ was created by ϕ and is contained in $\gamma(\sigma_1,\ldots,\sigma_p)$, the latter has also been created by ϕ at some point before $\varphi^\star$ starts. Using the same reasoning than for the converse balance property, by maximality of φ we deduce that $n = 0$ so that $\varphi(B,S) = (B',S)$ for all S. Now by lemma 4.3, φ is a wbp. Hence φ begins in a !-node, and thus by induction on the length of ϕ, there sits a !-cycle ψ', such that $\psi_1 = \gamma\varphi\psi'\varphi^\star\gamma^\star\psi_0$ is a suffix of ϕ and a b-string. Going on like this, will clearly yield a !-cycle ψ which is a suffix of ϕ, as announced. □

Call and return of a !-cycle.

Let φ be a wbp and ψ be a !-cycle starting and ending in an **of course** link `n`. Suppose that ψ is a subpath of φ. Necessarily φ crosses `n` upwardly so that by lemma 4.4, φ has a unique sub-wbp φ_1 connecting the root of an exponential tree t_1 to `n` such that $\varphi_1\psi$ is a subpath of φ. We call φ_1 the *call path* of ψ in φ (denoted $\mathrm{call}_\varphi(\psi)$). Furthermore, since φ doesn't end in `n`, there is an exponential branch γ_1 of t_1 such that $\gamma_1\varphi_1\psi$ is contained in φ. We call γ_1 the *opening branch* of ψ in φ (denoted $\mathrm{open}_\varphi(\psi)$).

Symmetrically there is a unique wbp φ_2 connecting the root of an exponential tree t_2 to `n` and an exponential branch γ_2 of t_2 such that $\psi\varphi_2^\star\gamma_2^\star$ is a subpath of φ. We call φ_2 the *return path* of ψ in φ (denoted $\mathrm{return}_\varphi(\psi)$) and γ_2 the *closing branch* of ψ in φ (denoted $\mathrm{close}_\varphi(\psi)$). Summing up, we have that φ contains the subpath:

$$\mathrm{open}_\varphi(\psi)\,\mathrm{call}_\varphi(\psi)\psi\,\mathrm{return}_\varphi(\psi)^\star\,\mathrm{close}_\varphi(\psi)^\star$$

for any !-cycle ψ contained in φ.

Legal paths.

A wbp φ connecting two multiplicative links is *legal* if for any !-cycle ψ contained in φ we have:

$$\mathrm{call}_\varphi(\psi) = \mathrm{return}_\varphi(\psi) \quad \text{and} \quad \mathrm{open}_\varphi(\psi) = \mathrm{close}_\varphi(\psi).$$

Theorem 4.7 (Legal and regular paths) *Let ϕ be a wbp; then ϕ is legal iff ϕ is regular, that is, $a(\phi)$ is not the empty map.*

Proof. Consider a moment $a(\phi)$ as acting on $B^{-1}S$, that is B reversed concatenated with S, instead of (B,S). With this notational variant one gets back the model of the geometry of interaction described in [1] which was shown in [2] to assign non empty maps to legal paths and only to them. $\square$

5 The jumping abstract machine

Analysis of legal runs.

As stated by the theorem above 4.7, IAM runs are exactly legal paths. Now the call-return symmetry of legal paths suggests that much too much computation is done by the IAM. More precisely let ϕ be a legal path, ψ be a !-cycle at some box b in ϕ, φ be the call (and return by legality) path of ψ, γ be the opening (and closing by legality) branch of ψ in some exponential tree t. So ϕ is $\ldots\gamma\varphi\psi\varphi^\star\gamma^\star\ldots$

Now the computation of the action $a(\gamma\varphi\psi\varphi^\star\gamma^\star)$ may be decomposed in:

(i) $\gamma(\sigma_1 : \cdots : \sigma_p : B, S) = (B, \sigma : S)$ where σ is an exponential signature depending on $\sigma_1,\ldots,\sigma_p$ and p is the lift of γ;

(ii) $\varphi(B,\sigma:S) = (\tilde{\varphi}(B),\sigma:S)$ because φ is a wbp, hence satisfies the balance property 4.1;

(iii) $\psi(\tilde{\varphi}(B),\sigma:S) = (\tilde{\varphi}(B),\sigma:\tilde{\psi}(S))$ because ψ is a !-cycle, hence satisfies the boxes property 4.5. Note that we assume here that B is chosen so that $\tilde{\varphi}(B)$ is in the domain of the partial identity π defined in lemma 4.5, for otherwise the computation stops here.

(iv) $\varphi^\star(\tilde{\varphi}(B),\sigma:\tilde{\psi}(S)) = (B,\sigma:\tilde{\psi}(S))$ because $\tilde{\varphi}$ is one-one.

(v) $\gamma^\star(B,\sigma:\tilde{\psi}(S)) = (\sigma_1:\cdots:\sigma_p:B,\tilde{\psi}(S))$.

At the beginning of step iv the legality condition imposes what is to follow: $\varphi^\star\gamma^\star$. Also note that right before that same step, σ which was the exponential signature built by γ is directly accessible on top of the balancing stack $\sigma:\tilde{\psi}(S)$.

Therefore, the action of $\varphi^\star\gamma^\star$ is only to move back to the starting node $\mathtt{n}$ of γ (a dereliction link by definition of exponential branches) and restore the boxes stack $\sigma_1:\cdots:\sigma_p:B$. Now if the action of γ at step i were to push the address of $\mathtt{n}$ together with the stack $\sigma_1:\cdots:\sigma_p:B$ on top of S, in place of the exponential signature σ, then at step iv one could pop this information which would be precisely on top of $\tilde{\psi}(S)$, in the balance stack, jump directly to $\mathtt{n}$ and restore $\sigma_1:\cdots:\sigma_p:B$, ready to continue the computation. This would save the computation of $\varphi^\star\gamma^\star$.

Optimization.

We shall now build an abstract machine which computes legal runs. The JAM proceeds by moving inside a net R, managing a *state* consisting in the couple of an *environment* B and a stack S. Objects stored into B and S are as before the constants P and Q, and additionally *closures* in place of signatures, i.e, pairs $(\mathtt{n},B)$ where $\mathtt{n}$ is the address of a dereliction link and B is an environment. Moves and their associated transitions on states are the same as the actions in the IAM except there are no more upward moves in **why not** links, and one has instead of the downward moves in an exponential branch and in an **of course**, the two expected alternative transitions, one that pushes a closure on S, and the other that pops it from B:

setjump (downward an exponential branch dereliction): let p be the lift of the exponential branch and $\mathtt{n}$ its starting **dereliction** node; then the transition is:

$$(\rho_1:\cdots\rho_p:B,S) \to (B,(\mathtt{n},\rho_1:\cdots\rho_p:B),S)$$

longjump (downward a principal door): the state has the form $(\rho:B,S)$ where ρ is a closure $(\mathtt{n},B')$, the transition jumps to the premise of the **dereliction** $\mathtt{n}$, changes the state to (B',S) and gets ready to move upward.

where $\mathtt{n}$ is the address of the starting dereliction and p is the lift of the exponential branch.

Notations.

Let $\mathtt{n}$ be (the address of) a dereliction link. Then $\mathtt{n}$ is a leaf of an exponential tree $t_{\mathtt{n}}$ and determines a unique exponential branch $\gamma_{\mathtt{n}}$ in $t_{\mathtt{n}}$. Let $p_{\mathtt{n}}$ be the lift of $\gamma_{\mathtt{n}}$. Then the action $a(\gamma_{\mathtt{n}})$ has the form:

$$\gamma_{\mathtt{n}}(\sigma_1 : \cdots : \sigma_{p_{\mathtt{n}}} : B, S) = (B, \sigma_{\mathtt{n}}(\sigma_1, \ldots, \sigma_{p_{\mathtt{n}}}) : S)$$

where $\sigma_{\mathtt{n}}$ is a one-one mapping associating an exponential signature to each $p_{\mathtt{n}}$-uple (not each pineapple) of exponential signatures.

Relation to the IAM.

We define inductively a mapping associating to each stack S (resp. environment B) of a state a balancing stack $\hat{S}$ (resp. box stack $\hat{B}$) and to each closure ρ an exponential signature $\hat{\rho}$:

- if S is $X : S'$ for a multiplicative constant X then $\hat{S}$ is $X : \hat{S'}$;
- if S is $\rho : S'$ for some closure σ then $\hat{S}$ is $\hat{\rho} : \hat{S'}$; same for B;
- if ρ is the closure $(\mathtt{n}, B)$ then, assuming the notations of the foregoing paragraph, B has the form $\rho_1 : \cdots : \rho_{p_{\mathtt{n}}} : B'$ and we define $\hat{\rho}$ to be $\sigma_{\mathtt{n}}(\hat{\rho}_1, \ldots, \hat{\rho}_{p_{\mathtt{n}}})$.

Note that $\hat{\rho}$ doesn't depend on B' above, and hence can be arbitrarily smaller than ρ. Thus the optimized process will need more space to manage its additional information.

Theorem 5.1 (Correctness of the JAM) *Let R be a net, with no exponential axioms, let $\mathtt{n}$ and $\mathtt{m}$ be two nodes of R. The JAM moves from $\mathtt{n}$ to $\mathtt{m}$, changing a state (B, S), with no closures in B nor S, into a state (B', S'), iff there is a path ϕ in R linking $\mathtt{n}$ to $\mathtt{m}$ and such that $\phi(\hat{B}, \hat{S}) = (\hat{B'}, \hat{S'})$.*

Which means that the JAM agrees with the IAM if no closures are given in advance. A particular and important case to apply the theorem is when $\mathtt{n}$ and $\mathtt{m}$ are conclusions of R. Then this means that, up to the $\hat{.}$ translation the J-machine computes the execution formula as the I-machine does.

Proof. The proof is by induction on the transitions of the JAM.

1. Suppose the J-run is above a premise of the principal door of a box b, preparing for a downward !-transition. Note that the boxes stack can't be empty because of the depth invariant, so that the transition always takes place and the J-run now jumps.

We only have to show that ϕ just completed a call/!-cycle configuration. Indeed, if this is the case, the analysis of legal runs at the beginning of this section proves that the IAM will run until it completes the return path and there arrive with the correct state.

First, there is a suffix of ϕ, say ψ which is a !-cycle, by lemma 4.6. Now we need to show there is a wbp, say φ, such that $\varphi\psi$ is a suffix of ϕ. Put $(B, \sigma : S)$

to be the I-state at the beginning of ψ, and define φ to be the longest path such that $\varphi\psi$ is a suffix of ϕ on the one hand, and on the other hand $\varphi^\star(B,S)$ is defined for *all* S. Note that, since σ was created by ϕ and by hypothesis is tranported by φ, the latter is a proper subpath of the former. Therefore by the maximality hypothesis and the same reasonning than along the converse balance lemma, φ must be a wbp.

2. In any other cases the argument is routine; the reader might try the case when the jump is set to check the definition of $\hat{\rho}$. □

6 A translation of λ-calculus

We shall now give a definition of nets derived from a translation of pure λ-calculus. We work with the formulae I, O, $?I$ and $!O$ which are asked to satisfy:

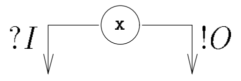
$$O^\perp = I \quad \text{and} \quad O = !O \multimap !O$$

The second equation, which allows interpretation of *pure* λ-calculus, is similar to the domain equation $D = !(D \multimap D)$ whose solutions are models of (call-by-value) λ-calculus. It is used by, *e.g.*, Abadi, Gonthier and Lévy in [2]. For the purpose of translating λ-calculus into nets, an other possible equation is $O = !O \multimap O$.

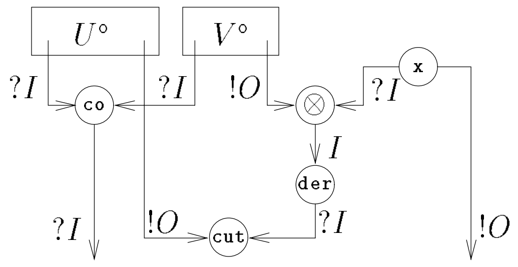
Translating λ-calculus.

To each λ-term T with free variables x_1, ..., x_n we associate a net T° with n conclusions $?I$ and one conclusion $!O$. We keep a one-one correspondence between the $?I$-conclusions of T° and the free variables of T along the translation:

Variable: a variable x is translated into an **axiom**:

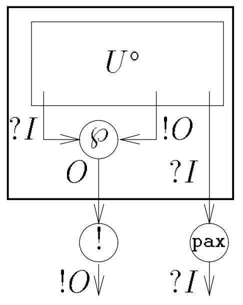


Application: if T is $(U)V$ then its translation is:



Each shared variables of U° and V° are contracted by means of **contraction** links.

Abstraction: the λ-abstraction corresponds to a **par** and a box, *i.e.*, if T is $\lambda x\, U$ then T° is:



where the $?I$ premise of the **par** is the conclusion of U° corresponding to x. If x doesn't occur in T, then the $?I$ premise of the **par** is created with a **weakening** link.

Correspondence between T and T°.

The translation is made in such a way that each edge typed with a $!O$ formula corresponds to a unique subterm of T: the $!O$ conclusion of R corresponds to the whole term, the $!O$ conclusion of an **axiom** link corresponds to either an occurrence of variable in T if the **axiom** was introduced by the first rule, or to an application subterm $(U)V$ in T if the **axiom** was introduced by the second rule. The $!O$ conclusion of an **of course** link corresponds to an abstraction subterm $\lambda x\, U$ in T.

Therefore, each **of course** link, as well as each **par** link, corresponds to a unique λ in T. Each **dereliction** link, each **tensor** link and each **cut** link in T° corresponds to a unique application in T. Moreover the leaves of the exponential trees in T° (if not the degenerate case of the **dereliction**'s) are all **axiom** links corresponding to occurrences of variable in T. Thus to each occurrence of variable in T corresponds a unique exponential path starting from the conclusion of an **axiom**.

De Bruijn code.

Consider an exponential path γ in T° attached to an occurrence of variable x in T and denote $\mathtt{a}_x$ the **axiom** link associated to x and $\mathtt{n}_x$ the root of the corresponding exponential tree. Let p_x be the lift of γ. By definition p_x is the number of boxes that γ exits when moving downward from $\mathtt{a}_x$ to $\mathtt{n}_x$. But each such box corresponds to an abstraction subterm of T which contains the occurrence x since $\mathtt{a}_x$ is contained in the boxes. Furthermore none of these abstractions can bind x since γ only contains **pax** links. Now if x is bounded in T then (the conclusion of) $\mathtt{n}_x$ is premise of a **par** link followed by an **of course** link, both corresponding to λx in T. In this case p_x is nothing but the *de Bruijn code* of x.

7 λ-calculus and the JAM

$!O$-transitions.

Let $R = T^\circ$ be a net obtained by the translation of a term T. We shall now describe all $!O$-*transitions* of the JAM, *i.e.*, the three compound moves of the JAM from a $!O$ formula upwardly to a $!O$ formula upwardly. In λ-calculus terms a $!O$-transition corresponds to a move from a subterm of T to a subterm of T.

We suppose that we start from the $!O$ conclusion of R with an empty state (empty environment and empty stack). Along the analysis we shall maintain two invariants (which are obviously satisfied at the beginning):

Depth invariant: B has the shape $\rho_1 : \cdots : \rho_m$ where m is the depth of the current position, *i.e.*, the number of boxes containing the current position, and the ρ_i's are closures;

Stack invariant: S has the shape: $S = \rho_1 : Q : \rho_2 : Q : \cdots : \rho_n : Q$ for some n, where the ρ_i are closures.

So suppose we are moving upwardly in a $!O$-edge e with state (B, S). There are three cases:

Application case:

e is the conclusion of an **axiom** link $\mathtt{a}$ which comes from the translation of an application in T. Then the moves to come are:

(i) down the $?I$-conclusion f of $\mathtt{a}$; since we are in the application case, f must be the right premise of a **tensor** link, therefore the move changes the state into $(B, Q : S)$;

(ii) down the I conclusion of the **tensor** which must be the premise of a **dereliction** link $\mathtt{d}$ so the state is now $(B, (\mathtt{d}, B) : Q : S)$;

(iii) down the $?I$ conclusion of $\mathtt{d}$ which is premise of a **cut** link. This move leaves the state invariant;

(iv) up the $!O$ premise of the **cut** link, which again leaves the state invariant and finishes the sequence with state $(B, (\mathtt{d}, B) : Q : S)$.

Since this sequence of moves never crossed a door of a box, the depths of the initial and final links are the same, thus the depth invariant is respected. Obviously the stack invariant is respected too.

Abstraction case:

e is the conclusion of an **of course** link coming from the translation of a λ in T. If the stack S is empty the machines does nothing. Otherwise the sequence of moves is:

(i) up the premise f of the **of course** link which is conclusion of a **par** link. Since S is $\rho : Q : S'$ for some closure ρ and some stack S' the move changes the state into $(\rho : B, Q : S')$.

(ii) now we are to move up a premise of the **par** link; since the state is $(\rho : B, Q : S)$ we choose the right premise which is typed by $!O$, and we stop in its source link with state $(\rho : B, S')$.

In this $!O$-transition we entered one box so the depth increased by one. Also the length of the environment increased by one since we popped a closure from the stack into the environment. Thus the depth invariant was preserved. Since the sequence popped the two first elements of S, the stack invariant also is.

Variable case:

e is the $!O$ conclusion of an **axiom** link $\mathtt{a}$ which this time comes from the translation of an occurrence of variable x. Let γ be the exponential path associated to x, $\mathtt{n}_\gamma$ its final link and p_γ its lift. By definition of lift, the depth of $\mathtt{a}$ is at least p_γ, thus by the depth invariant we have $B = \rho_1 : \cdots : \rho_{p_\gamma} : B'$ for some B' and some closures ρ_i. The $!O$-transition is:

(i) down γ; since the lift of γ, *i.e.*, the number of **pax** crossed by γ is p_γ, the effect of this sequence of moves is to change the state into (B', S);

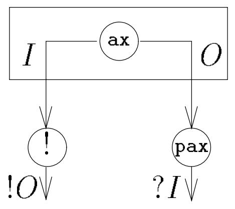
(ii) if x is free in T then $\mathtt{n}_\gamma$ is a conclusion of T° and the machine stops there. Otherwise the conclusion f of $\mathtt{n}_\gamma$, which is typed by $?I$, is the left premise of the **par** link corresponding to λx. Moving downward f changes the state into $(B', P : S)$;

(iii) the conclusion of the **par** must be the premise of an **of course** link, *i.e.*, the principal door of a box. By the depth invariant again we have $B' = \rho : B''$ where ρ is the closure $(\mathtt{d}, B_0)$. Thus the transition is to jump to the premise g of the **dereliction** link $\mathtt{d}$ with state $(B_0, P : S)$;

(iv) g must be the conclusion of a **tensor** link. Because of the P on top of the stack, the next move is to go up the *left* premise of the **tensor** which is typed by $!O$. Therefore this move ends the $!O$-transition with state (B_0, S).

Since S is invariant during this $!O$-transition, the stack invariant is respected. On the other hand, since the initial state is empty, ρ was created at some previous step s of the execution by a downward move in the premise of $\mathtt{d}$. Since B_0 was stored in ρ, this means that at step s, the state had the shape (B_0, S') for some S'. Therefore by induction on s, which is strictly earlier than the current step, the depth invariant was respected so that B_0 has the right number of elements.

Note that there is a small gap here. Indeed we defined the J-machine for nets with atomic axioms and we are using it with nets with non atomic axioms (conclusions are $!O$, $?I$). To fill the gap we have added to the J-machine some transitions allowing us to move downward an exponential path step by step: in contraction do nothing, in **pax** pop the first closure from B. In fact this addition to the J-machine can be simulated if one η-expands the non atomic axioms, that is replacing all non atomic axioms links by:



To be completely precise, the reader can check that the new J-machine acting on non atomic nets save some jumps: when the old machine acting on the η-expanded net was making a series of jumps through η-expanded axioms starting from an **of course** link coming from a λ and ending into a **dereliction** coming from an application, the new one makes only one jump, as described above.

Conclusion.

A $!O$-position in the net T° may be encoded by its corresponding subterm in T. Also, if the **dereliction** link d corresponds to the application subterm $(U)V$ of T, one may encode the address of d by V. This is unambiguous since there is a one-one correspondence between **dereliction** links in T° and application subterms of T, as said before. With these new conventions the $!O$-transitions are simply the KAM's transitions, hence:

Theorem 7.1 (JAM/KAM-isomorphism) *The JAM, when applied to λ-terms translated as in section 6 is isomorphic to the KAM.*

References

[1] Martin Abadi, Georges Gonthier, and Jean-Jacques Lévy. The geometry of optimal lambda reduction. In *Proceedings of the 19th ACM Symposium on Principles of Programming Languages*, pages 15–26, 1992.

[2] Andrea Asperti, Vincent Danos, Cosimo Laneve, and Laurent Regnier. Paths in the lambda-calculus. In *Proceedings of the 9th Symposium on Logic in Computer Science*. IEEE Computer Society Press, 1994.

[3] Vincent Danos. *Une Application de la Logique Linéaire à l'Étude des Processus de Normalisation (principalement du λ-calcul)*. Thèse de doctorat, Université Paris 7, 1990.

[4] Vincent Danos, Hugo Herbelin, and Laurent Regnier. Games semantics and abstract machines. In *Proceeding of the 11th Symposium on Logic in Computer Science*. IEEE Computer Society Press, 1996.

[5] Vincent Danos and Laurent Regnier. The structure of multiplicatives. *Archive for Mathematical Logic*, 28, 1989.

[6] Vincent Danos and Laurent Regnier. Local and asynchronous beta-reduction. In *Proceedings of the 8th Symposium on Logic in Computer Science*. IEEE Computer Society Press, 1993.

[7] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[8] Jean-Yves Girard. Geometry of interaction I: an interpretation of system F. In Ferro & al., editor, *Proceedings of A.S.L. Meetings*. North-Holland, 1988.

[9] Jean-Yves Girard. Geometry of interaction II: Deadlock free algorithms. In Martin-Löf & Mints, editor, *Proceedings of COLOG'88*, volume 417 of *Lecture Notes in Computer Science*, pages 76–93. Springer-Verlag, 1988.

[10] Jean-Yves Girard. Quantifiers in linear logic II. In *Nuovi problemi della logica e della filosofia della scienza, CLUEB*, Bologna, 1991.

[11] Jean-Yves Girard. Geometry of interaction III: acommodating the additives. In Jean-Yves Girard, Yves Lafont, and Laurent Regnier, editors, *Advances in Linear Logic*, volume 222 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, 1995.

[12] Yves Lafont. From proof-nets to interaction nets. In Jean-Yves Girard, Yves Lafont, and Laurent Regnier, editors, *Advances in Linear Logic*, volume 222 of *London Mathematical Society Lecture Note Series*, pages 225–247. Cambridge University Press, 1995.

[13] Ian Mackie. The geometry of implementation. In *Proceedings of the 22th ACM Symposium on Principles of Programming Languages*, 1995.

[14] Laurent Regnier. *Lambda-Calcul et Réseaux*. Thèse de doctorat, Université Paris 7, 1992.

[15] Laurent Regnier. Une équivalence sur les lambda-termes. *Theoretical Computer Science*, 126, 1994.