Light Logics and Optimal Reduction: Completeness and Complexity

Patrick Baillot*
LIPN, CNRS & Université Paris Nord
pb@lipn.univ-paris13.fr

Paolo Coppola†
Università di Udine
coppola@uniud.it

Ugo Dal Lago‡
Università di Bologna
dallago@cs.unibo.it

Abstract

Typing of lambda-terms in Elementary and Light Affine Logic (EAL, LAL resp.) has been studied for two different reasons: on the one hand the evaluation of typed terms using LAL (EAL resp.) proof-nets admits a guaranteed polynomial (elementary, resp.) bound; on the other hand these terms can also be evaluated by optimal reduction using the abstract version of Lamping's algorithm. The first reduction is global while the second one is local and asynchronous. We prove that for LAL (EAL resp.) typed terms, Lamping's abstract algorithm also admits a polynomial (elementary, resp.) bound. We also show its soundness and completeness (for EAL and LAL with type fixpoints), by using a simple geometry of interaction model (context semantics).

1 Introduction

Background and Motivations. Light logics such as Light Affine Logic (LAL) and Elementary Affine Logic (EAL) have been introduced in [20, 1] and then studied as type systems for lambda-calculus [5, 12, 13, 6, 9, 11] and semantically (*e.g.* in [25, 28, 16]). Their analysis has been motivated by two distinct features:

1. *Complexity-certified reduction*: using the syntax of *proof-nets*, the logics LAL (EAL, respectively) ensure that the program terminates with a polynomial (elementary, respectively) time bound;
2. *Simple optimal reduction*: for lambda-terms typed in these systems one can use the abstract version of Lamping's algorithm [24], without the so-called *oracle*, so plain graph rewriting with local rules.

However each of these approaches has its drawbacks:

- Proof-net reduction is global, requires the handling of *boxes*, and thereby enforces a form of synchronicity which is awkward to implement;
- Optimal reduction is local, asynchronous – and elegant, but . . . does not offer any guaranteed complexity bound.

Actually, even the fact that Lamping's abstract algorithm is *correct* for terms of EAL and LAL is not completely straightforward: it was pointed out since [1] and is part of the folklore, but it seems no explicit direct proof of this statement is available in the literature, although the general techniques of [22] could be applied to this restricted setting.

The goal of this paper is therefore to bring together the best of these two worlds: we prove that terms typable in LAL (EAL, respectively) can be reduced by Lamping's abstract algorithm with a certified polynomial (elementary, respectively) time bound. Moreover a type derivation in LAL or EAL carries two kinds of information: the sharing information and the boxing information. We actually show here that the boxing information is *not* needed to perform Lamping's abstract algorithm. Some systems like DLAL [9] or restrictions of EAL [12] do not use sharing: in that case knowing that the term is typeable (without knowing the type) is sufficient to apply the abstract algorithm.

Actually the bounds of light logics can also be obtained without the proof-net machinery, in plain lambda-calculus, if one considers fragments of the type systems, possibly with restricted (lazy) reduction [9, 11]. However this is still a global form of reduction (β-reduction). Here we aim to handle the full type systems and to switch to a local reduction, which is motivating for concrete implementations and in particular for distributed evaluation [29].

Optimal Reduction and Light Logics. The fact that EAL typable terms can be reduced with Lamping's abstract algorithm is quite remarkable, since it is known that the bookkeeping needed for the oracle causes inefficiencies in optimal reduction [26].

On the other hand, as proof-net reduction in these systems is performed with guaranteed complexity bound, one might think that the preservation of bounds when switching from proof-net reduction to optimal reduction is a conse-

*Partially supported by projects NO-CoST (ANR, JC05_43380), CRISS (ACI).

†Partially supported by a visiting professorship of Univ. Paris Nord.

‡Work partially carried out while the author was postdoctoral fellow of Univ. Paris Nord and of Univ. Denis Diderot, Paris 7.

quence of optimality itself. However this is actually not true: the optimality concerns the number of parallel beta-steps, which is not directly related to the normalisation time [4, 2]. For an in-depth study of optimal reduction one can consult [3].

Moreover, techniques used when analyzing proof-net (or lambda-term) reduction time cannot be directly applied here. In particular, the level-by-level reduction strategy (see [20, 5]) has no counterpart in the framework of sharing graphs, where copying is done incrementally.

Contributions. Our main results are:

- We define a general class of admissible translations from light logics type derivations to sharing graphs, subsuming the ones proposed before.
- For each admissible translation, we show that graph reduction is sound and complete with respect to lambda-reduction.
- Moreover, we show that graph reduction can be performed in bounded time, where the bound is of the same complexity order as the one we have on the underlying logical system.

Moreover we believe that the main technique used to prove the complexity bounds (Section 6), based on the definition of *weights* for sharing graphs (or interaction nets, [23]) following [14] is of its own interest and could presumably be applied to other problems.

A full version with complete proofs is available [7].

2 Soundness and Completeness in the General Case

Before introducing the specific logical systems we are interested in, we define the notions of soundness and completeness for abstract systems of graph reduction. Throughout the paper, Λ is the set of pure, untyped, lambda terms. If A is a set and $\rightarrow$ is a binary relation on A, the set of normal forms in A (with respect to $\rightarrow$) will be denoted $NF(A)_{\rightarrow}$.

Definition 1 (Graph Rewriting Systems) *A Θ-graph rewriting system is a quintuple $(\Theta, \Delta, \rightarrow_{\Delta}, \mathcal{T}, \mathcal{R})$ where:*

- *$\Theta \subseteq \Lambda$ is a set of lambda-terms to which the technique can be applied.*
- *Δ is a set of* graphs.
- *$\rightarrow_{\Delta}$ is a rewriting relation on Δ.*
- *$\mathcal{T}$ is a total binary relation from Θ to Δ, called the* initial translation.
- *$\mathcal{R}$ is a function from Δ to Λ, called the* readback.

Note that $\mathcal{T}$ is a relation and not a mere function, since we want to allow several possible translations of a term (this is related to the fact that we will allow the possibility to decorate a given lambda-term as several different proof-nets).

$$\begin{tikzcd} t \arrow[r, dashed, "*"] \arrow[d, "\mathcal{T}"'] & u \\ G \arrow[r, "*", "\Delta"'] & H \arrow[u, "\mathcal{R}"'] \end{tikzcd} \qquad \begin{tikzcd} t \arrow[r, "*"] \arrow[d, "\mathcal{T}"'] & u \\ G \arrow[r, dashed, "*", "\Delta"'] & H \arrow[u, "\mathcal{R}"'] \end{tikzcd}$$

Figure 1. Soundness and Completeness

Definition 2 (Soundness) *We say that the Θ-graph rewriting system $(\Theta, \Delta, \rightarrow_{\Delta}, \mathcal{T}, \mathcal{R})$ is* sound *with respect to a reduction relation $\rightarrow$ on Λ iff for every term $t \in \Theta$, if $G \in \mathcal{T}(t)$ and G reduces to normal form H (in $\rightarrow_{\Delta}$) then t reduces to normal form u (in $\rightarrow$) and $\mathcal{R}(H) = u$.*

Soundness of a Θ-graph rewriting system implies that if we start with a term t in Θ, translate it into a graph, reduce the graph and finally read-back a term u, then u is the normal form of t. This does not mean the Θ-graph rewriting system will necessarily do its job: to be sure about that, we need completeness:

Definition 3 (Completeness) *We say that the Θ-graph rewriting system $(\Theta, \Delta, \rightarrow_{\Delta}, \mathcal{T}, \mathcal{R})$ is* complete *with respect to a reduction relation $\rightarrow$ on Λ iff for every term $t \in \Theta$ if t reduces to normal form u, then any $G \in \mathcal{T}(t)$ reduces to normal form H, where $\mathcal{R}(H) = u$.*

3 Type Assignment Systems and Proof-Nets

Formulae of (intuitionistic) *elementary affine logic* (EAL for short) are generated by the following productions:

$$A ::= \alpha \mid A \multimap A \mid !A \mid \forall\alpha.A \mid \mu\alpha.A$$

where α ranges over a countable set of *atoms*. Recall that ! is called an *exponential connective* or *modality*.

Here we are considering in fact EAL with type fixpoints (recursive types) but this does not modify its normalisation properties [15]. Most references in the literature deal with the second order fragment $\mathsf{EAL}_{\forall}$, which does not include type fixpoints.

EAL can be seen as a type system for terms in Λ: Figure 2 presents type assignment by means of sequent calculus, which is tedious for typing but convenient for studying the dynamics. Other presentations of typing can be found in the literature [12, 13, 10]. Note that sharing is allowed, for instance by using rules X and U. Θ_{EAL} denotes the set of lambda terms which are typable in elementary affine logic.

Elementary affine logic proofs can be formulated as a system of (intuitionistic) proof-nets Δ_{EAL}, defined inductively on Figure 3. Node X is called a *contraction* node. The *principal edge* of a node v is the edge incident to v through its *principal port* (indicated with a $\bullet$). A *cut* is an edge $e = \{v, w\}$ which is principal for both v and w.

EAL proof-nets can be endowed with a rewriting relation $\rightarrow_{\mathsf{EAL}}$ that we do not report here for lack of space (see [14]).

Axiom, Cut and Structural Rules

$$\frac{}{x:A\vdash x:A}\,A \qquad \frac{\Gamma\vdash t:A \quad \Delta,x:A\vdash u:B}{\Gamma,\Delta\vdash u\{t/x\}:B}\,U$$

$$\frac{\Gamma\vdash t:B}{\Gamma,x:A\vdash t:B}\,W \qquad \frac{\Gamma,x:!A,y:!A\vdash t:B}{\Gamma,z:!A\vdash t\{z/x,z/y\}:B}\,X$$

Multiplicative Logical Rules

$$\frac{\Gamma,x:A\vdash t:B}{\Gamma\vdash\lambda x.t:A\multimap B}\,R_{\multimap} \qquad \frac{\Gamma\vdash t:A \quad \Delta,x:B\vdash u:C}{\Gamma,\Delta,y:A\multimap B\vdash u\{yt/x\}:C}\,L_{\multimap}$$

Exponential Rules

$$\frac{\Gamma\vdash t:A}{!\Gamma\vdash t:!A}\,P_!$$

Second Order Rules

$$\frac{\Gamma\vdash t:A \quad \alpha\notin FV(\Gamma)}{\Gamma\vdash t:\forall\alpha.A}\,R_{\forall} \qquad \frac{\Gamma,x:A\{B/\alpha\}\vdash t:C}{\Gamma,x:\forall\alpha.A\vdash t:C}\,L_{\forall}$$

Least Fixpoint Rules

$$\frac{\Gamma\vdash t:A\{\mu\alpha.A/\alpha\}}{\Gamma\vdash t:\mu\alpha.A}\,R_{\mu} \qquad \frac{\Gamma,x:A\{\mu\alpha.A/\alpha\}\vdash t:B}{\Gamma,x:\mu\alpha.A\vdash t:B}\,L_{\mu}$$

Figure 2. A sequent calculus for elementary linear logic with second order and fixpoints.

The important case of $\rightarrow_{\mathsf{EAL}}$ is when an X node meets a $R_!$ node, corresponding to a box: in this case the box is duplicated and the doors $L_!$ of the two copies are linked to X nodes (*contraction normalisation step*).

If v (resp. e) is a node (resp. edge) of a proof-net, $\partial(v)$ (resp. $\partial(e)$) denotes its *box-depth* (*level*). If $G\in\Delta_{\mathsf{EAL}}$ is a proof-net, its depth $\partial(G)$ is the maximal depth of its edges. The *stratification property* of EAL states that the depth $\partial(e)$ of an edge does not change through $\rightarrow_{\mathsf{EAL}}$.

Light Affine Logic. LAL can be obtained from EAL by adopting a stricter exponential discipline: one restricts the rule $P_!$ of EAL to the case where Γ contains at most one formula, but also adds a new connective § with rule $P_\S$ (see Figure 4). The connective § is a weak form of !, that does not allow for contraction (rule X).

There is a translation $(.)^e$ from LAL to EAL formulae obtained by replacing § with !. It extends to a translation on proofs.Therefore the set Θ_{LAL} of lambda-terms typable in LAL is included in Θ_{EAL}.

The proofs-nets of LAL are defined as those of EAL but with two new nodes $L_\S$ and $R_\S$ and conditions on boxes: a box with $R_!$ main door (!-box) can have at most one $L_!$ door; a box with $R_\S$ main door (§-box) can have any number of $L_\S$ and $L_!$ doors. A rewriting relation $\rightarrow_{\mathsf{LAL}}$ is defined on these proof-nets [14]. This reduction does not cause any duplication of a §-box.

The translation $(.)^e$ can be extended naturally to a translation from LAL to EAL proof-nets, and it maps $\rightarrow_{\mathsf{LAL}}$ to $\rightarrow_{\mathsf{EAL}}$. Therefore the set of LAL proof-nets can be seen as a subset of Δ_{EAL}. Hence properties of EAL proof-nets will

$$\frac{\vdash t:A}{\vdash t:!A}\,P_!^1 \qquad \frac{x:A\vdash t:A}{x:!A\vdash t:!A}\,P_!^2 \qquad \frac{\Gamma,\Delta\vdash t:A}{\S\Gamma,!\Delta\vdash t:\S A}\,P_\S$$

Figure 4. Exponential rules of light affine logic with second order and fixpoints.

be valid in particular for LAL proof-nets and we will state them only for EAL (except for complexity issues in Section 6).

Paths. A *direct path* is a sequence of edges $e_1,\ldots,e_n$ such that the following conditions hold:

- For every $1\leq i<n$, e_i and e_{i+1} have a vertex v_i in common;
- For every $1\leq i<n$, $e_i\neq e_{i+1}$ and either e_i or e_{i+1} is principal for v_i.

An example of a direct path is reported in Figure 5(a). We say that a direct path $e_1,\ldots,e_n$ with $n\geq 2$ *starts* at v iff $e_1=\{v,w\}$ is principal for v and there is z with $e_2=\{w,z\}$. A direct path $e_1,\ldots,e_n$ is *simple* iff for every $1\leq i<n$, the edge e_{i+1} is principal for v_i. The direct path in Figure 5(b) is simple, while the one in Figure 5(a) is not. A direct path is *maximal* iff it is not part of any longer direct path. A box b in a proof-net N is *special* iff any direct path starting from one of its premises is simple.

Lemma 1 *Any non-simple direct path $e_1,\ldots,e_n$ starting at any node v contains a cut e_i such that $\partial(e_i)\leq\partial(e_1)$.*

Strategies. There are two reduction strategies for proof-nets in Δ_{EAL} (or Δ_{LAL}) that are of particular interest for our purposes:

- The *level-by-level strategy*, LBL. A cut at level $n+1$ cannot be reduced if there are cuts at level n.
- The *modified level-by-level strategy*, MLBL. It is the level-by-level one with an additional constraint: whenever we copy a box b, b must be a special box.

The fact that MLBL is a reduction strategy is a consequence of Lemma 1 and of the fact that proof-nets satisfy the correctness criterion [20].

Complexity Bounds. We can now recall the main results on EAL and LAL :

Theorem 1 (Girard [20]) *For every natural number n, there is a polynomial (respectively, elementary function) $e_n:\mathbb{N}\rightarrow\mathbb{N}$ such that for every proof-net N of LAL (respectively, EAL) if $N\rightarrow^n M$ in the MLBL strategy, then $n\leq e_{\partial(N)}(|N|)$ and $|M|\leq e_{\partial(N)}(|N|)$.*

Recall that binary lists can be represented in LAL with the type: $W=\forall\alpha.!(\alpha\multimap\alpha)\multimap!(\alpha\multimap\alpha)\multimap\S(\alpha\multimap\alpha)$. This way, a term of LAL type $t:W\multimap\S^kW$ can be converted to a proof-net, and its application to a list evaluated in

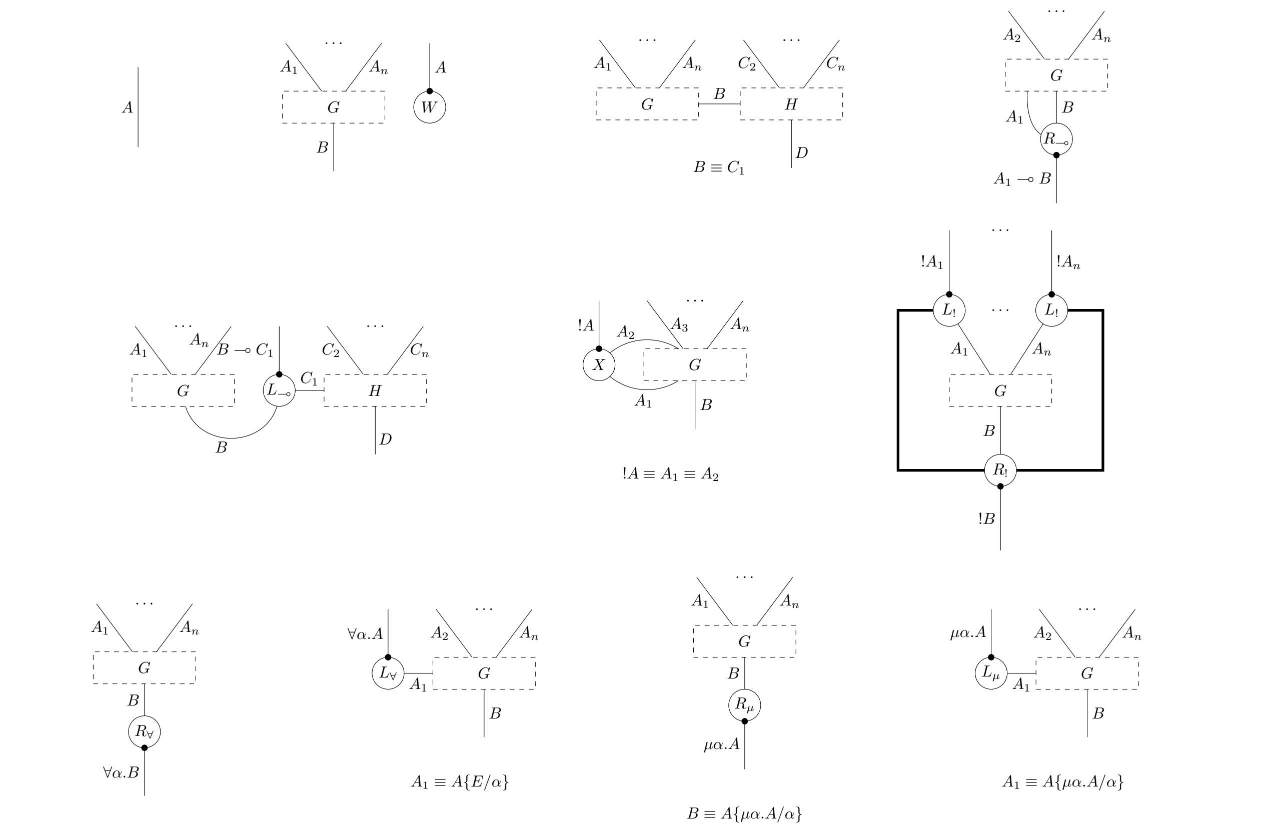
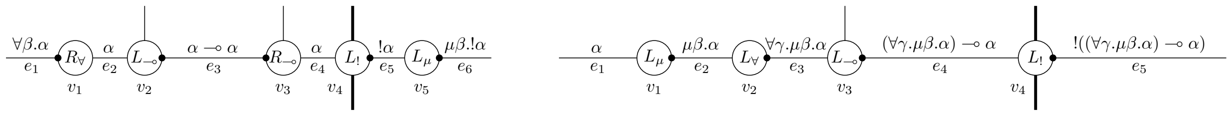


Figure 3. Proof-nets for elementary affine logic.



(a) (b)

Figure 5. Paths: some examples

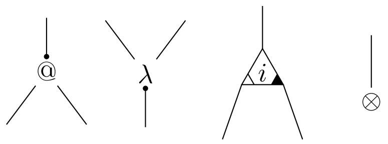


Figure 6. Sharing Nodes.

polynomial time using $\rightarrow_{\mathsf{LAL}}$. However this is still a global evaluation procedure and we want to replace it by optimal reduction.

4 Lamping's Abstract Algorithm

Now we turn to the local reduction procedure. *Abstract sharing graphs* in Δ_{ASR} are defined by the nodes of Figure 6: the 3rd node is called *fan* and is given together with an integer *index* i. A rewriting relation $\rightarrow_{\mathsf{ASR}}$ on Δ_{ASR} is defined on Figure 7. Notice that we omit the garbage collection rules. This omission is anyway harmless: the readback procedure is not affected by the presence of garbage and the complexity of garbage collection is linear in the size of the graph.

If G is a sharing graph, $fp(G)$ is the set of its *free ports* (dangling edges) and $wp(G)$ that of edges incident to $\otimes$-nodes. If u is a node, then $pp(u)$ is its principal port. Principal ports for lambda and application nodes are marked with a point in Figure 6. The principal port of the fan is the unmarked one.

To translate proof-nets into sharing graphs we need to set the indices for the fans. For that, any proof-net N is given together with a *labelling function* $\mathcal{F}$ from the set of its contraction nodes to natural numbers. We require the function $\mathcal{F}$ for the proof-net N to be *compatible with depths*: if $\mathcal{F}(v) = \mathcal{F}(w)$ then $\partial(v) = \partial(w)$. The translation $\mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}$ is defined up to such labelling functions: $\mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(N, \mathcal{F})$ is the graph $G \in \Delta_{\mathsf{ASR}}$ obtained in the following way:

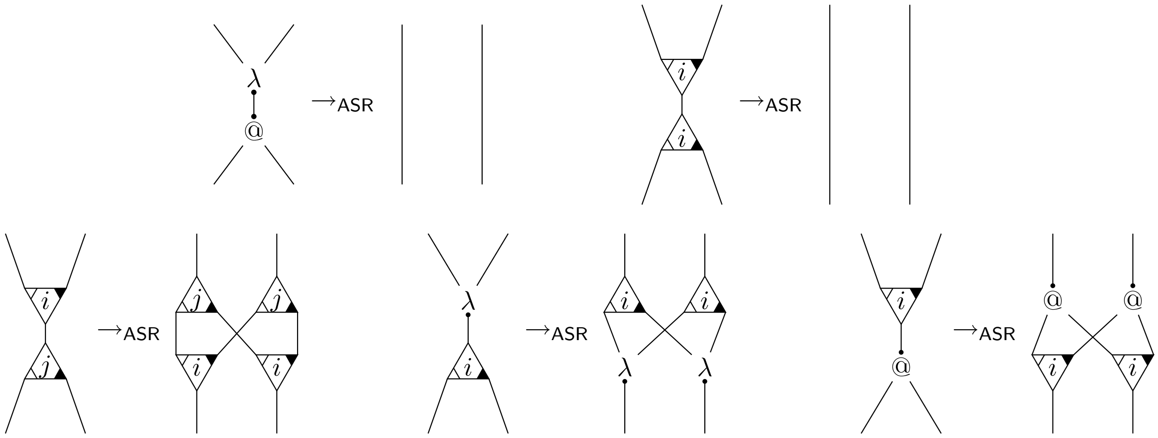


Figure 7. Rewriting rules for sharing graphs.

- Replace nodes $R_{\multimap}$ (resp. $L_{\multimap}$) by nodes λ (resp. @),
- Remove boxes and nodes $L_!$, $R_!$, $L_\forall$, $R_\forall$, L_μ, R_μ,
- Replace each contraction node v with a fan with index $\mathcal{F}(v)$.

We denote by $|\mathcal{F}|$ the cardinality of the image of the labelling function $\mathcal{F}$.

Note that in a proof-net reduction step $N \rightarrow_{\mathsf{EAL}} M$, each node of M comes from a unique node of N; a labelling function $\mathcal{F}$ for N then induces in a natural way a labelling function for M, that we will also write $\mathcal{F}$. By the stratification property of EAL, if $\mathcal{F}$ is compatible with depths for N, then so it is for M.

In previous works on light logics and optimal reduction, two particular translations of proof-nets have been considered:

- The *level translation*, LT: the labelling function $\mathcal{F}$ is the one defined by the depths, that is $\mathcal{F}(v) = \partial(v)$.
- The *distinct labelling translation*, DLT: $\mathcal{F}$ assigns a distinct index to each contraction node.

Note that the second translation does not need the information provided by boxes in N. The first translation, on the other hand, has the advantage that it minimizes the number of indices used to annotate fans. The properties that we will prove in the rest of the paper are valid for all these translations. We give on Fig.8(a) an abstract sharing graph that will serve as running example. It is obtained as the DLT of a proof-net corresponding to a derivation of $f :!(A \multimap A) \multimap !(A \multimap A) \multimap B, g :!(A \multimap A) \vdash u : B$, where $u = (\lambda x.f\ x\ x)(\lambda z.g\ z)$. In Fig. 8(b) we give its normal form for $\rightarrow_{\mathsf{ASR}}$.

The concepts of principal port, direct path, simple path, etc. can be easily transferred from proof-nets to sharing graphs. The number of maximal paths in a cut-free sharing graph is bounded:

Lemma 2 *Let G be a cut-free sharing graph and let e be one of its free ports. Then there are at most $|G|+1$ maximal direct paths in the form $e = e_1, \ldots, e_n$.*

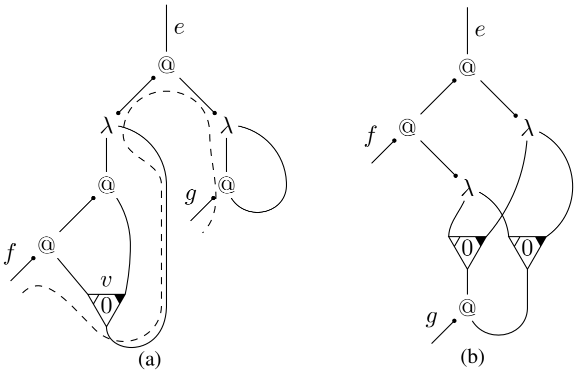


Figure 8. Example.

Now we want to bound the complexity of this rewriting procedure and show that it is sound and complete.

5 Context Semantics

Context semantics will be the tool for showing soundness of sharing graph reduction (following [21]). A *context* can be seen as a token carrying a piece of information and travelling around the net [18]. As we are considering a more constrained setting than [21, 19, 17] the contexts can be presented as tuples, as in [8]. This reflects the stratified structure of EAL proof-nets.

Definition 4 (Elementary contexts) *An* elementary context *C of length k is a tuple of stacks $S_1, \ldots, S_k, T$ over the alphabet $\{\mathsf{p}, \mathsf{q}\}$. Stacks S_i are called* exponential stacks*, stack T is called* multiplicative stack.

We denote by ε the empty stack, xS the result of pushing x on S and by SS' the concatenation of S and S'. Let $\sqsubseteq$ be the prefix ordering on stacks. We also denote by $\sqsubseteq$ the pointwise order on the product of stacks. Finally $\sqsubseteq_m$ will denote the order on elementary contexts defined by identity on the exponential stacks S_i and $\sqsubseteq$ on the multiplicative stack T.

Definition 5 (Valid paths) *Let N be in Δ_{EAL} and $\mathcal{F}$ a labelling function, with $k = |\mathcal{F}|$.*

- *A* context *of N relative to $\mathcal{F}$ is a pair (p, C) where p is an edge of N and C is an elementary context of length $k+1$.*
- *The binary relation $\sim$ on contexts is defined by symmetrically closing the rules in Table 1 and adding for the other (binary) nodes the rule acting as identity on the elementary context (no rule for the W node).*
- *A direct path $e_1, \ldots, e_n$ in N is* valid *with respect to two elementary contexts C_1 and C_n iff the nodes along the path transform the context (e_1, C_1) into (e_n, C_n). More precisely, there must be elementary contexts $C_2, \ldots, C_{n-1}$ such that $(e_i, C_i) \sim (e_{i+1}, C_{i+1})$ whenever $1 \leq i < n$. Then we write $(e_1, C_1) \triangleright (e_n, C_n)$ and say the path $e_1, \ldots, e_n$ is* persistent.

Definition 6 (Context semantics) *Given a proof-net N of EAL and a labelling function $\mathcal{F}$, the context semantics $[\![N]\!]_{\mathcal{F}}$ of N is the set of pairs $((e, C), (f, D))$ such that e and f are conclusion edges of N and $(e, C) \triangleright (f, D)$.*

We will sometimes write simply $[\![N]\!]$ instead of $[\![N]\!]_{\mathcal{F}}$. Notice that as the transitions are deterministic and as when reaching a conclusion no transition is possible anymore, if $(e, C) \triangleright (f, D)$ and $(e, C) \triangleright (g, E)$ are both in $[\![N]\!]_{\mathcal{F}}$ then $f = g$ and $D = E$. Therefore the context semantics of N induces a (partial) function on contexts. Note, however, that there can be two essentially different reasons why the context semantics is undefined on (e, C):

- The current context could get "stuck" at some node: there is no possible transition,
- The current context could travel indefinitely without reaching a conclusion.

However we will see in Section 5.2 that this second possibility is guaranteed never to occur for proof-nets.

Given a sharing graph G and a partition $\mathcal{F}$ of its fan nodes, its contexts and context semantics $[\![G]\!]$ are defined similarly to that of proof-nets (Table 2). It is then clear that the context semantics is preserved by the translation from proof-nets to sharing graphs:

Proposition 1 *Let N be an EAL proof-net and $\mathcal{F}$ a partition of its contraction nodes, then $[\![N]\!]_{\mathcal{F}} = [\![\mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(N, \mathcal{F})]\!]$.*

We give some examples of contexts in the context semantics of the sharing graph from Fig.8(a):

$$(f, \varepsilon, \mathsf{pq}) \triangleright (g, \mathsf{p}, \mathsf{q}); \quad (f, \varepsilon, \mathsf{qpq}) \triangleright (g, \mathsf{q}, \mathsf{q});$$
$$(e, \varepsilon, \varepsilon) \triangleright (f, \varepsilon, \mathsf{qq}); \quad (g, \mathsf{p}, \mathsf{p}) \triangleright (f, \varepsilon, \mathsf{pp});$$
$$(g, \mathsf{q}, \mathsf{p}) \triangleright (f, \varepsilon, \mathsf{qpp}).$$

The path corresponding to the first of these contexts is represented on Fig.8(a) by a dashed line.

If P is a set of contexts, P^- denotes the subset of P including only minimal elements (with respect to $\sqsubseteq$). When traversing any node in a sharing graph G, only one particular stack of the underlying context can be modified. Two nodes u and v have the same *sort* (formally, $ty(u) = ty(v)$) iff they can modify the same stack. For instance @ and λ nodes have the same sort. Given a node u, $ep(u)$ is the set of contexts whose stack corresponding to u is ε.

Lemma 3 (Monotonicity) *Suppose $e_1, \ldots, e_n$ is a direct path valid with respect to $C_1, \ldots, C_k, T$ and $D_1, \ldots, D_k, S$. Moreover, suppose that $E_1, \ldots, E_k, U$ are stacks. Then $e_1, \ldots, e_n$ is valid with respect to $C_1E_1, \ldots, C_kE_k, TU$ and $D_1E_1, \ldots, D_kE_k, SU$.*

5.1 Reduction and Context Semantics

Now we consider the behaviour of the context semantics with respect to the reduction of proof-nets and sharing graphs. Let us start with the latter case, which is easier.

Since sharing graph reduction is local, all valid paths starting and ending with a free port are preserved during reduction. Consequently:

Proposition 2 *Let G be a sharing graph and $G \to_{\mathsf{ASR}} H$ then $[\![G]\!] = [\![H]\!]$.*

As to proof-nets the situation is more delicate. It is well-known that geometry of interaction or context semantics are usually not preserved by general proof-net reduction [19, 17, 27]. To deal with this problem we define a partial order $\preccurlyeq$ on context functions. Context semantics will be preserved up to $\succcurlyeq$ but that will be sufficient to obtain a soundness result with respect to lambda-calculus.

Definition 7 *Let f, g be two partial functions on contexts. Then $f \preccurlyeq g$ iff for any context p, C we have:*

1. *If $f(p, C)$ is defined, then $g(p, C) = f(p, C)$,*
2. *If $f(p, C)$ is undefined then either:*
   i. *$g(p, C)$ is undefined,*
   ii. *or $f(p, D)$ is undefined whenever $D \sqsupseteq_m C$.*

The point in 2.ii is that $f(p, C)$ is undefined, but it is not merely because of a lack of information in the multiplicative stack, since no increase of information on this stack can trigger an answer. The behaviour of f on such input is in fact irrelevant for the read-back process that we will define.

Lemma 4 *The relation $\preccurlyeq$ is a partial order.*

Now we have:

Proposition 3 *Let N be an EAL proof-net and $N \to_{\mathsf{EAL}} M$ then $[\![N]\!] \succcurlyeq [\![M]\!]$.*

Table 1. Context Semantics for Proof-nets

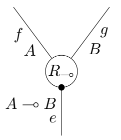
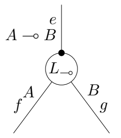
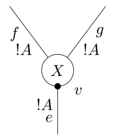
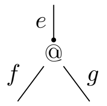
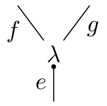
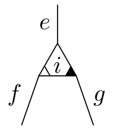
	$(e,(S_1,\dots,S_{\lvert\mathcal{F}\rvert},\mathsf{p}T)) \sim (f,(S_1,\dots,S_{\lvert\mathcal{F}\rvert},T))$ $(e,(S_1,\dots,S_{\lvert\mathcal{F}\rvert},\mathsf{q}T)) \sim (g,(S_1,\dots,S_{\lvert\mathcal{F}\rvert},T))$
	$(e,(S_1,\dots,S_{\lvert\mathcal{F}\rvert},\mathsf{p}T)) \sim (f,(S_1,\dots,S_{\lvert\mathcal{F}\rvert},T))$ $(e,(S_1,\dots,S_{\lvert\mathcal{F}\rvert},\mathsf{q}T)) \sim (g,(S_1,\dots,S_{\lvert\mathcal{F}\rvert},T))$
	$(e,(S_1,\dots,S_{\mathcal{F}(v)-1},\mathsf{p}S_{\mathcal{F}(v)},S_{\mathcal{F}(v)+1},\dots,S_{\lvert\mathcal{F}\rvert},T)) \sim (f,(S_1,\dots,S_{\lvert\mathcal{F}\rvert},T))$ $(e,(S_1,\dots,S_{\mathcal{F}(v)-1},\mathsf{q}S_{\mathcal{F}(v)},S_{\mathcal{F}(v)+1},\dots,S_{\lvert\mathcal{F}\rvert},T)) \sim (g,(S_1,\dots,S_{\lvert\mathcal{F}\rvert},T))$

Table 2. Context Semantics for Sharing Graphs

	$(e,(S_1,\dots,S_n,\mathsf{p}T)) \sim (f,(S_1,\dots,S_n,T))$ $(e,(S_1,\dots,S_n,\mathsf{q}T)) \sim (g,(S_1,\dots,S_n,T))$
	$(e,(S_1,\dots,S_n,\mathsf{p}T)) \sim (f,(S_1,\dots,S_n,T))$ $(e,(S_1,\dots,S_n,\mathsf{q}T)) \sim (g,(S_1,\dots,S_n,T))$
	$(e,(S_1,\dots,S_{i-1},\mathsf{p}S_i,S_{i+1},\dots,S_n,T)) \sim (f,(S_1,\dots,S_n,T))$ $(e,(S_1,\dots,S_{i-1},\mathsf{q}S_i,S_{i+1},\dots,S_n,T)) \sim (g,(S_1,\dots,S_n,T))$

5.2 Acyclicity

We now describe properties of valid paths in the proof-nets and sharing graphs we are dealing with.

Proposition 4 (Finiteness of valid paths for proof-nets) *Let N be an EAL proof-net. Then there exists an integer k such that for any valid path $e_1,\dots,e_n$ we have $n \leq k$.*

Proof. This is proved in [8] for $\mathsf{EAL}_\forall$, and the proof can be easily adapted to EAL using the fact that EAL is strongly normalising. □

A *cycle* is a direct path $e_1,\dots,e_n$ with $n \geq 2$ such that $e_1 = e_n$ and :

- $e_1,\dots,e_n$ is valid with respect to $C = C_0,\dots,C_k,T$ and $D = D_0,\dots,D_k,S$;
- For every $0 \leq i \leq k$, either $C_i \sqsubseteq D_i$ or $D_i \sqsubseteq C_i$;
- Either $T \sqsubseteq S$ or $S \sqsubseteq T$.

Proposition 5 (Acyclicity of Proof-Nets) *If N is a proof-net, then its context semantics does not contain any cycle.*

Proof. Indeed if the proof-net N contained a cycle, then by repeatedly composing it with itself one would get valid paths of arbitrary length, which would contradict Proposition 4. □

Proposition 6 *Let N be an EAL proof-net, $G = \mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(N,\mathcal{F})$ and $G \to^*_{\mathsf{ASR}} H$. Then there exists an inte-*

ger k such that: for any valid path $e_1, \ldots, e_n$ of H we have $n \leq k$.

Proof. [sketch] The statement holds for the paths of G itself because of Prop. 4 and of the fact that any valid path of G can be lifted back to a valid path of N of same length or longer. Then for H obtained from G by one step of $\rightarrow_{\mathsf{ASR}}$, one can show that if H has valid paths of arbitrary length, then so has G, which yields a contradiction. □

6 Complexity

We study the complexity of sharing graph reduction by defining a weight W_G for any sharing graph G. The underlying idea is the following: the weight of G is the sum of the individual weight of each $u \in V_G$, the latter being the number of possible copies of u that are produced during normalisation. We will relate the weight to the number of reduction steps of the sharing graph, and then, for sharing graphs coming from EAL (and LAL), bound the weight by using the properties of proof-nets.

Formally, the weight of an edge u will be defined as the number of different valid paths $e_1, \ldots, e_n$ satisfying certain additional constraints. First of all, e_1 must be $pp(u)$. Secondly, e_n must be:

- Either the principal edge of a node v such that $ty(u) = ty(v)$.
- Or an edge in $\mathit{fp}(G) \cup wp(G)$.

This way the weight of u will be exactly the number of copies of u that will eventually appear during reduction of G. This can be characterized by context semantics :

Definition 8 (Weight) *Let $G \in \Delta_{\mathsf{ASR}}$. Then:*

- *If u is a node of G, then B_u, P_u and E_u are sets of elementary contexts defined in Figure 9.*
- *The weight W_G of G is defined as follows:*

$$W_G = \sum_{u \in V_G} (|B_u^-| + |P_u^-| + |E_u^-| - 1).$$

Notice that W_G can be either a natural number or ω.

Notice how 1 is subtracted from the sum $|B_u^-|+|P_u^-|+|E_u^-|$ when defining W_G. This way, W_G always decreases at any copying normalisation step, as we will see.

As an example: the weight of the graph of Fig.8(a) is $W_G = 2$, because all seven nodes have exactly one path of the kind considered (either counting for B_u^- or P_u^-) except the fan node and the rightmost λ node (binding z) who both have two paths, yielding $|B_u^-| = 2, \quad |P_u^-| = 0$.

The weight of a cut-free sharing graph obtained by reducing another sharing graph coming from a proof-net is always null:

Lemma 5 *If $N \in \Delta_{\mathsf{EAL}}$ and $\mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(N, \mathcal{F}) \rightarrow^*_{\mathsf{ASR}} G$ where $G \in \Delta_{\mathsf{ASR}}$ is a cut-free graph, then $W_G = 0$.*

Proof. Consider any $u \in V_G$ and any direct path starting from u. This path is always simple, since we assume G to be cut-free. Moreover, by Proposition 6, we cannot go on forever building up the path. As a consequence, we will end up at an edge in $\mathit{fp}(G) \cup wp(G)$. This, in particular, implies that $|P_u^-| + |E_u^-| = 1$, while $|B_u^-| = 0$. The thesis follows easily. □

Lemma 6 *If $N \in \Delta_{\mathsf{EAL}}$ is a cut-free proof-net and $\mathcal{F}$ is any partition of its contraction nodes, then $W_{\mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(N,\mathcal{F})} = 0$.*

Now, observe that annihilation rewriting steps leave W_G unchanged (while $|G|$ decreases by 2), whereas copying rewriting steps make W_G decrease by 2 (while $|G|$ increases by 2). As a consequence:

Proposition 7 *If $N \in \Delta_{\mathsf{EAL}}$, $G = \mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(N, \mathcal{F})$, W_G is finite and $G \rightarrow^n_{\mathsf{ASR}} H$, then $n \leq W_G + |G|/2$ and $|H| \leq W_G + |G|$.*

Given a proof-net $N \in \Delta_{\mathsf{EAL}}$ such that $N \rightarrow_{\mathsf{EAL}} M$, we can study the difference $W_{\mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(N,\mathcal{F})} - W_{\mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(M,\mathcal{F})}$. In particular, in the case of the MLBL strategy, the difference can be tightly bounded, because the number of paths that we "lose" at each reduction step can be properly bounded by an appropriate function (with the same order of magnitude as the one from Theorem 1) applied to $|N|$. This implies the weight of the underlying sharing graph cannot decrease too much during MLBL proof-net reduction and, by Lemma 6 and Theorem 1, we get:

Proposition 8 *For every natural number n, there is an elementary function $e_n : \mathbb{N} \rightarrow \mathbb{N}$ such that for every proof-net $N \in \Delta_{\mathsf{EAL}}$, $W_{\mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(N)} \leq e_{\partial(N)}(|N|)$.*

A similar result holds for LAL:

Proposition 9 *For every natural number n, there is a polynomial $e_n : \mathbb{N} \rightarrow \mathbb{N}$ such that for every proof-net $N \in \Delta_{\mathsf{LAL}}$, $W_{\mathcal{T}^{\mathsf{LAL}}_{\mathsf{ASR}}(N)} \leq e_{\partial(N)}(|N|)$.*

By Propositions 9 and 8, we get:

Theorem 2 *For every natural number n, there is a polynomial (resp. elementary function) $e_n : \mathbb{N} \rightarrow \mathbb{N}$ such that for every term t typable in LAL (resp. EAL), if N is a proof-net corresponding to a type derivation of t, then any reduction of the sharing graph $\mathcal{T}^{\mathsf{LAL}}_{\mathsf{ASR}}(N)$ (resp. $\mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(N)$) has length bounded by $e_{\partial(N)}(|N|)$.*

As an application, if t can be typed in LAL with type $W \multimap \S^k W$ then there exists a polynomial p such that the application of t to the term representing the list w, reduced using sharing graphs, takes at most $p(|w|)$ steps.

$$
\begin{array}{rcl}
B_u & = & \big\{ C \mid \exists v, D.\big((pp(u), C) \triangleright (pp(v), D)\big) \wedge (C \in ep(u)) \wedge (D \in ep(v)) \wedge (ty(u) = ty(v)) \big\} \\
P_u & = & \big\{ C \mid \exists q, D.\big((pp(u), C) \triangleright (q, D)\big) \wedge (C \in ep(u)) \wedge (q \in fp(G)) \big\} \\
E_u & = & \big\{ C \mid \exists q, D.\big((pp(u), C) \triangleright (q, D)\big) \wedge (C \in ep(u)) \wedge (q \in wp(G)) \big\}
\end{array}
$$

Figure 9. Definition of B_u, P_u and E_u.

7 Soundness

Suppose we are in the following situation:

$$
\begin{tikzcd}
t \arrow[r, "*"] \arrow[d, "\mathcal{T}"] & u \\
G \arrow[r, "*"] & H
\end{tikzcd}
$$

In words, we translated a typable term t to a sharing graph G, then normalised G to H. We now need to define a read-back procedure $\mathcal{R}$ that extracts the normal form u of t from H. We have to design a variant of the readback procedures in the literature, *e.g.* [21, 27], since here we are not handling a generic encoding of terms into proof-nets but an encoding based on type derivations.

The procedure $\mathcal{R}_\Lambda^{\mathsf{ASR}}$ is defined on sharing graphs, but does not look at the internal structure of the graph itself; rather, the procedure is defined as a set of *queries* to the underlying context semantics. To prove $\mathcal{R}_\Lambda^{\mathsf{ASR}}$ is correct, we can suppose the input graph to come from a cut-free proof-net M. This is enough to prove soundness. Indeed, the context semantics of M is essentially the same as the one of H:

$$
\begin{array}{ccc}
[\![N]\!] & \succcurlyeq & [\![M]\!] \\
 & & \| \\
[\![G]\!] & = & [\![H]\!]
\end{array}
$$

Observe that we could even apply the readback procedure to G, without reducing G to its normal form H. This, however, would make the read-back process less efficient, as the whole computation would be done by the context semantics.

We do not present the details of the read-back procedure here (see [7]). However, we sketch how it can be applied to the sharing graph (in normal form) in Figure 8(b), obtaining the normal form t of $(\lambda x.fxx)(\lambda z.gz)$. First of all, observe that the edge e corresponds to the root of the term, while edges f and g correspond to the free variables of same name. The read-back can be performed as follows:

- First of all, we want to know the head variable of t. To do that, we query the context semantics on $(e, \varepsilon, \varepsilon)$, obtaining $(e, \varepsilon, \varepsilon) \triangleright (f, \varepsilon, \mathsf{qq})$. This tells us that the head variable of t is f (since we end up in f) and that it is applied to two arguments u and v (since the rightmost stack is qq). The context corresponding to (this occurrence of) f is $(f, \varepsilon, \varepsilon)$.
- Now, we can ask ourselves what is the head variable of u. To do that, we query the context semantics on $(f, \varepsilon, \mathsf{p})$. It is undefined, meaning that the head variable (occurrence) we are looking for lies in the scope of a lambda abstraction. We try with $(f, \varepsilon, \mathsf{pq})$ obtaining $(f, \varepsilon, \mathsf{pq}) \triangleright (g, \mathsf{p}, \mathsf{q})$. This implies the head variable of u is g (since we end up in g) and that it is applied to one argument, call it s. Moreover, the context corresponding to this occurrence of g is $(g, \mathsf{p}, \varepsilon)$.
- The next question is: which is the head variable of s? We proceed in the usual way, querying the context semantics on $(g, \mathsf{p}, \mathsf{p})$ and $(g, \mathsf{p}, \mathsf{p}) \triangleright (f, \varepsilon, \mathsf{pp})$. Remembering that $(f, \varepsilon, \varepsilon)$ is the context corresponding to the head variable of t, we can infer that the head variable of s is bound by the first lambda-abstraction appearing in u (namely the first argument to f in t). Moreover, this variable occurrence has not any argument. So, we can conclude $t = f(\lambda z.gz)v$.
- Similarly, we can query the context semantics to get some information on v and obtain $v = (\lambda z.gz)$.

This query-protocol can be generalized into a function $\mathcal{R}_\Lambda^{\mathsf{ASR}} : \Delta_{\mathsf{ASR}} \rightarrow \Lambda$ whose value on a graph G only depends on the context semantics of G. But proving $\mathcal{R}_\Lambda^{\mathsf{ASR}}$ to be correct on sharing graphs in the form $\mathcal{T}_{\mathsf{ASR}}^{\mathsf{EAL}}(N)$ where N is cut-free is relatively easy. Thus, we get:

Theorem 3 (Soundness) *The Θ_{EAL}-graph rewriting system $(\Theta_{\mathsf{EAL}}, \Delta_{\mathsf{ASR}}, \rightarrow_{\mathsf{ASR}}, \mathcal{T}_{\mathsf{ASR}}^{\mathsf{EAL}}, \mathcal{R}_\Lambda^{\mathsf{ASR}})$ is sound.*

8 Completeness

Theorem 4 (Completeness) *The Θ_{EAL}-graph rewriting system $(\Theta_{\mathsf{EAL}}, \Delta_{\mathsf{ASR}}, \rightarrow_{\mathsf{ASR}}, \mathcal{T}_{\mathsf{ASR}}^{\mathsf{EAL}}, \mathcal{R}_\Lambda^{\mathsf{ASR}})$ is complete.*

Proof. It is sufficient to observe that Theorem 2 implies that reducing G will lead to a normal form H. Then it follows from the soundness result of Section 7 that $\mathcal{R}_\Lambda^{\mathsf{ASR}}(H) = u$. $\square$

9 Conclusions

We proved that Lamping's abstract algorithm is sound and complete for beta reduction of EAL (and LAL) typable pure lambda-terms. Moreover, the number of graph interaction steps is shown to be bounded by the same kind of bounds holding for proof-net normalisation. All these results have been established by exploiting context semantics. In particular, complexity results have been inferred in an innovative way, being inspired by [14].

Further work includes the extension of the approach to general optimal reduction. In the full algorithm, however, relative bounds should take the place of absolute bounds, since any pure lambda term can be reduced.

Acknowledgements. We are grateful to Harry Mairson and Simone Martini for stimulating discussions and advice on optimal reduction. We also wish to thank Vincent Atassi for useful conversations.

References

[1] A. Asperti. Light Affine Logic. In *Proc. of LICS 1998*, pages 300–308. IEEE Computer Society, 1998.

[2] A. Asperti, P. Coppola, and S. Martini. (Optimal) duplication is not elementary recursive. *Information and Computation*, 193:21–56, 2004.

[3] A. Asperti and S. Guerrini. *The Optimal Implementation of Functional Programming Languages*. Cambridge University Press, 1998.

[4] A. Asperti and H. Mairson. Parallel beta reduction is not elementary recursive. *Information and Computation*, 170(1):49–80, 2001.

[5] A. Asperti and L. Roversi. Intuitionistic light affine logic. *ACM Transactions on Computational Logic*, 3(1):1–39, 2002.

[6] V. Atassi, P. Baillot, and K. Terui. Verification of Ptime reducibility for system F terms via Dual Light Affine Logic. In *Proc. of CSL 2006*, volume 4207 of *LNCS*, pages 150–166. Springer, 2006.

[7] P. Baillot, P. Coppola, and U. Dal Lago. Light logics and optimal reduction: Completeness and complexity. Extended Version. Available at `http://www.arxiv.org/abs/0704.2448`, 2007.

[8] P. Baillot and M. Pedicini. Elementary complexity and geometry of interaction. *Fundamenta Informaticae*, 45(1-2):1–31, 2001.

[9] P. Baillot and K. Terui. Light types for polynomial time computation in lambda-calculus. In *Proc. of LICS 2004*, pages 266–275. IEEE Computer Society, 2004.

[10] P. Baillot and K. Terui. A feasible algorithm for typing in Elementary Affine Logic. In *Proc. of TLCA 2005*, volume 3461 of *LNCS*, pages 55–70. Springer, 2005.

[11] P. Coppola, U. Dal Lago, and S. Ronchi Della Rocca. Elementary affine logic and the call-by-value lambda calculus. In *Proc. TLCA 2005*, volume 3461 of *LNCS*, pages 131–145. Springer, 2005.

[12] P. Coppola and S. Martini. Optimizing optimal reduction. a type inference algorithm for elementary affine logic. *ACM Transactions on Computational Logic*, 7(2):219–260, 2006.

[13] P. Coppola and S. Ronchi Della Rocca. Principal typing for lambda calculus in elementary affine logic. *Fundamenta Informaticae*, 65(1-2):87–112, 2005.

[14] U. Dal Lago. Context semantics, linear logic and computational complexity. In *Proc. of LICS 2006*, pages 169–178. IEEE Computer Society, 2006.

[15] U. Dal Lago and P. Baillot. Light affine logic, uniform encodings and polynomial time. *Mathematical Structures in Computer Science*, 16(4):713–733, 2006.

[16] U. Dal Lago and M. Hofmann. Quantitative models and implicit complexity. In *Proc. of FSTTCS 2005*, volume 3821 of *LNCS*, pages 189–200. Springer, 2005.

[17] V. Danos and L. Regnier. Proof-nets and Hilbert space. In *Advances in Linear Logic*, pages 307–328. Cambridge University Press, 1995.

[18] V. Danos and L. Regnier. Reversible, irreversible and optimal lambda-machines. *Theoreritcal Computer Science*, 227(1-2):79–97, 1999.

[19] J.-Y. Girard. Geometry of interaction 1: interpretation of system F. In *Proc. Logic Colloquium 1988*, pages 221–260, 1989.

[20] J.-Y. Girard. Light linear logic. *Information and Computation*, 143:175–204, 1998.

[21] G. Gonthier, M. Abadi, and J.-J. Lévy. The geometry of optimal lambda reduction. In *Proc. of POPL 1992*, pages 15–26. ACM, 1992.

[22] S. Guerrini, S. Martini, and A. Masini. Coherence for sharing proof-nets. *Theoretical Computer Science*, 294(3):379–409, 2003.

[23] Y. Lafont. Interaction combinators. *Information and Computation*, 137(1):69–101, 1997.

[24] J. Lamping. An algorithm for optimal lambda calculus reduction. In *Proc. of POPL 1990*, pages 16–30. ACM, 1990.

[25] O. Laurent and L. Tortora de Falco. Obsessional cliques: a semantic characterization of bounded time complexity. In *Proc. of LICS 2006*, pages 179–188. IEEE Computer Society Press, 2006.

[26] J. L. Lawall and H. G. Mairson. Optimality and inefficiency: What isn't a cost model of the lambda calculus? In *Proc. of ICFP 1996*, pages 92–101. ACM, 1996.

[27] H. Mairson. From Hilbert spaces to Dilbert spaces: Context semantics made simple. In *Proc. of FSTTCS 2002*, volume 2556 of *LNCS*, pages 2–17, 2002.

[28] A. S. Murawski and C.-H. L. Ong. Discreet games, light affine logic and ptime computation. In *Proc. of CSL 2000*, volume 1862 of *LNCS*, pages 427–441. Springer, 2000.

[29] M. Pedicini and F. Quaglia. A parallel implementation for optimal lambda-calculus reduction. In *Proc. of PPDP 2000*, pages 3–14. ACM, 2000.