January 1997

Proof nets, Garbage, and Computations

Stefano Guerrini
University of Pennsylvania

Simone Martini
University of Pennsylvania

Andrea Masini
University of Pennsylvania

University of Pennsylvania Institute for Research in Cognitive Science Technical Report No. IRCS-97-02.

Proof nets, Garbage, and Computations

Abstract
We study the problem of local and asynchronous computation in the context of multiplicative exponential linear logic (MELL) proof nets. The main novelty is in a complete set of rewriting rules for cut-elimination in presence of weakening (which requires garbage collection). The proposed reduction system is strongly normalizing and confluent.

Comments
University of Pennsylvania Institute for Research in Cognitive Science Technical Report No. IRCS-97-02.

Institute for Research in Cognitive Science

Proof nets, Garbage, and Computations

Stefano Guerrini
Simone Martini
Andrea Masini

University of Pennsylvania
3401 Walnut Street, Suite 400A
Philadelphia, PA 19104-6228

January 1997

Site of the NSF Science and Technology Center for Research in Cognitive Science

IRCS Report 97--02

Proof nets, Garbage, and Computations*

S. Guerrini[1], S. Martini[2], A. Masini[3]

[1] IRCS, University of Pennsylvania,
3401 Walnut Street, Suite 400A, Philadelphia – USA;
stefanog@saul.cis.upenn.edu.
[2] Dipartimento di Matematica e Informatica, Università di Udine,
Via delle Scienze, 206, I-33100 Udine – Italy; martini@dimi.uniud.it.
[3] Dipartimento di Informatica, Università di Pisa,
Corso Italia, 40, I-56125 Pisa – Italy; masini@di.unipi.it.

Abstract: We study the problem of local and asynchronous computation in the context of multiplicative exponential linear logic (MELL) proof nets. The main novelty is in a complete set of rewriting rules for cut-elimination in presence of weakening (which requires garbage collection). The proposed reduction system is strongly normalizing and confluent.

Keywords: linear logic; typed lambda-calculus; cut-elimination; sharing graphs; proof nets.

1 Introduction

Cut-elimination (or normalization) is the logical description of the computational process of (term) reduction, central to most of the literature on lambda-calculus and related functional languages. From the pioneering description of beta-reduction in terms of normalization of natural deduction proofs (dating back to the sixties, by Curry, Prawitz, and Howard), this logical interpretation has been extended to a variety of functional languages and formal logical systems.

The arrival on the scene of linear logic [Gir87] gave a good momentum to this research area, for the stress on resource conscious computations. Using the key idea that the type constructor for the function space may be understood as a modal operator (explaining the process of duplicating and/or erasing input data) followed by a linear function, it was discovered soon that typed and untyped lambda-calculus may be faithfully embedded into linear logic, thus allowing the use of linear logic computations (in the form of proof net cut-elimination) to perform (or study) lambda-reduction. A crucial step was the discovery [GAL92] that Lamping's graph-reduction algorithm [Lam90] for optimal lambda-reduction (in the sense of Lévy, [Lév78]) could be interpreted as a way of performing proof net cut-elimination in a distributed and local way. The (global) concept of proof net box is replaced with information distributed on the graph (brackets, croissants, and indices). Cut-elimination is performed

* Partially supported by: HCM Project CHRX-CT93-0046 and CNR GNSAGA (Guerrini,Martini); BRA 8130 LOMAPS (Masini).

with a set of completely local graph-rewriting rules, main part of which are those manipulating the information added to the graph to (dynamically) reconstruct the boxes. The (potential) sharing (expressed with new nodes) of common subgraphs is the key to optimal reduction. Cut-elimination in these *sharing graphs* is based on three main ideas. First, in the reduction of a logical cut involving duplication of information, the duplication is not actually performed. It is instead indicated in a lazy way by the introduction of specific new nodes (*fans*). Second, new reduction rules are added to incrementally perform the required duplication. Third, there is a mechanism to recognize when this process of incremental and distributed duplication is over.

Sharing graphs have been revisited from different perspectives: a categorical interpretation (and new notation) [Asp95]; their extension to other logical systems [AL96]; their relations to the geometry of interaction [ADLR94]; a new notation ensuring better properties (in particular, that the normal form of a sharing graph be a proof net) [GMM96].

All these approaches differ in the specific way the bookkeeping information is coded into the sharing graph. However, they agree on their focus on what in [GMM96] we called *restricted* proof nets: weakening is not allowed. There are at least two reasons for this. First, the problems weakening raises during the reduction of an arbitrary proof net do not show up during the reduction of a lambda-term (better: of the proof net corresponding to a lambda-term), even if weakening *is* allowed in the term. Second, the usual syntax for proof nets do not seem to allow for *any* solution to those problems (see Section 2).

We propose in this paper a set of completely local and asynchronous graph rewriting rules for cut-elimination in proof nets for the Multiplicative Exponential Linear Logic (with weakening and contraction but without constants), MELL. The proposed rules are proved strongly normalizing and confluent. Moreover, the normal form of a sharing graph is a proof net. This generalizes to MELL the results of [GMM96].

As in [GMM96], these results rely on a sharp distinction between logic and control. In standard sharing graphs, the nodes used to control the reduction process (fans, brackets, and croissants) have in fact also a static role, to introduce logical formulas. In our approach, instead, new information, in the form of indexes over formulas, are responsible for the static correctness (that is, for *logic*), while the *control* nodes (muxes) are responsible only for the duplication and reindexing during cut-elimination. This logic vs. control separation is rooted in our previous work on indexed systems for linear logic [MM95].

To treat weakening we exploit a well known *permutability* result: In the sequent calculus formulation of MELL, the weakening rule permutes with all the other rules, and hence it can be pushed upwards, to the axioms. Axioms may then be formulated as

$$\vdash p, p^{\perp}, ?\Gamma,$$

dropping an explicit weakening rule. When expressed in a suitable proof net setting, this idea always generates connected proof nets, allowing a local graph-rewriting cut-elimination. The approach may be seen as a specialization of that

of Banach [Ban95].

In our setting, the cut-elimination of a box against a weakening may be performed in two (ideal) phases: first, a marking of the box to erase, keeping intact its logical structure; second, the actual erasing of the box, with the reorganization of its (secondary) doors as weakenings.

The structure of the paper is as follows. Section 2 discusses the problems weakening raises, informally introducing the techniques used in the sequel. Section 3 sets the stage with definitions and the relations with more usual formulations of proof nets. Section 4 introduces the rewriting rules. Section 5 states the main properties of the reduction systems. In Section 6 we discuss some relations with optimality and indicate further research lines. Proofs, or even intelligible sketches of them, are well beyond the space limits of this paper.

2 Weakening in Proof Net Reduction

Weakening in linear logic can produce boxes whose contents are disconnected, and, more subtly, such boxes can be generated by the cut-elimination procedure, even starting from proof nets whose boxes are connected. The crucial case (for cut-elimination) is that of a box whose principal door has as premise a weakening link. A more general situation is depicted in Figure 1 (left) (weakening boxes are not shown). The net σ is a correct proof net. The dotted region on the left is built starting from some weakenings and provides the principal door of a box. In a sequent proof, we would first construct the proof σ; then we would proceed with the weakenings; finally we would build π. We may called the dotted region comprising π a *weakening isle*: it is a separate connected component of the net; it is *not* a proof net by itself; the global correctness of the net is thus guaranteed by the presence of the proof net σ.

Cut now the principal door of the box against a contraction, as shown in Figure 1 (right). The reduction of the cut consists of the (global) duplication of the box, and the replacement of the cut with two cuts. But in sharing graphs boxes are not explicit. They may be reconstructed by means of the auxiliary information (brackets, indices, etc.) *through a graph exploration starting from their principal door*. And then we are lost. No matter which rules will be devised to rewrite the cut, if these rules are to be completely local, the σ part of the graph will never be affected by them and, hence, will not be duplicated. The results of any local rewriting of Figure 1 (right), then, cannot be much different from the graph shown in Figure 2. That graph is not the intended reduct of the cut; moreover is not a proof net, and it cannot be made into one by adding exponential boxes. There are two weakening isles, while only one copy of the proof net σ, which cannot validate both.

To solve the connected components problem we change the definition of the axiom link. Beside the dual atoms p and $p^{\perp}$, we attach to an axiom link also a list of weakening formulas. There is no explicit weakening link. In this way a proof net is always connected and there is hope for a local exploration of its boxes. The reduction of the cut of Figure 1 (b) may now be done in the standard

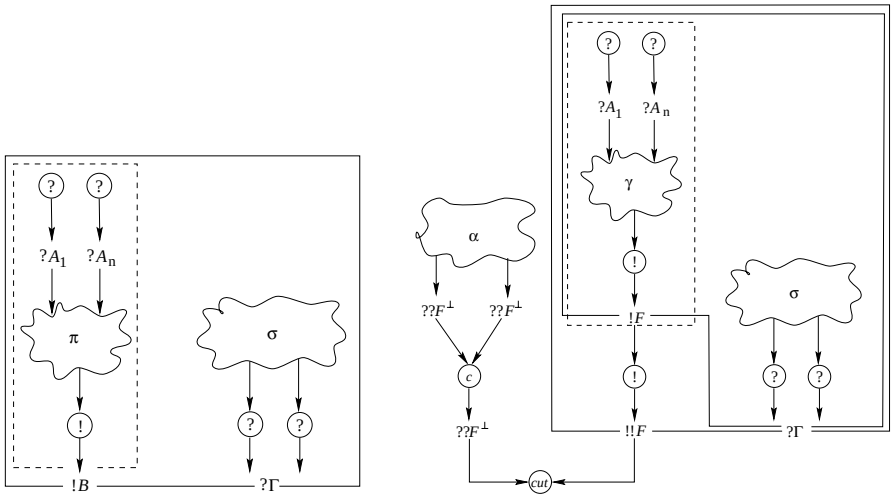


Figure 1. A weakening isle

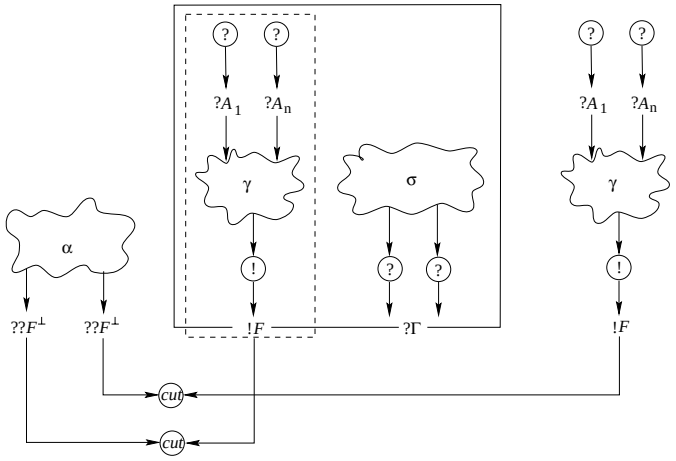


Figure 2. A mistake

sharing graph way: duplicate (and move) the cuts and add a link (a fan, or a *mux* in our terminology) indicating the sharing of the two boxes. The actual duplication will be done incrementally, making the mux travel inside the box. Since any weakening formula is explicitly connected to some axiom, the mux will eventually visit all the box, ensuring the duplication of σ.

This formulation of weakening in proof nets is a variant of the technique introduced by Banach [Ban95]. To prove the so-called sequentialization theorem

(that an acyclic and connected proof-structure comes indeed from a sequent linear logic proof and it is thus a proof net) for MELL, Banach introduced the notion of *probe*, an arc pointing "back" from a weakening link to any other link of the net, thus guaranteeing connectedness. In the example of Figure 1 (a), a probe would connect each ? link on top left to one link (anyone is fine) of σ. However, this approach remains too liberal (in the choice of the target link of a probe) for the purpose of a distributed cut-elimination rewriting algorithm. In fact, this freedom is not necessary. Our formulation forces the target of a probe to always be an axiom contained in the same weakening box.

Besides the "weakening isle" case, the other important situation involving weakening is that of a weakening formula (with its probe connecting to an axiom) cut against the principal door of a box, whose reduction is the erasing of the box and the "relocation" of its secondary doors into weakenings. In our approach this will happen in two ideal phases. First (*mark*), weakening and cut are replaced with a mux (connected through the probe to an axiom), which will explore the box, marking the links for deletion, but preserving the logical structure. This mux will stop its marking at the border of the box, like any other mux. Second (*sweep*), starting from the marked axioms, the box will be erased, reducing it to a special "garbage collector" link, which will collect all the secondary doors of the box. At the end, these secondary doors will be transformed into weakenings, with probes toward the axiom connected to the original weakening.

3 Proof Nets and ℓ-nets

3.1 Leveled Structures

We introduce the basic concepts we will use in the paper. The basic notion is that of sharing ℓ-structure, a box-free representation of (shared) proof structures.

Definition 3.1 (sℓ-structure). An *sℓ-structure* (sharing leveled structure of links) is a finite connected hypergraph whose nodes are labeled with indexed formulas (either a MELL formula or the extra-logical constant $\varnothing$ (*dummy*), decorated with a natural number, the *level* of the formula) and whose hyperedges (also called *links*) are labeled from the set $\{\mathsf{cut}, \mathsf{ax}, \wp, \otimes, !, ?, \bullet\} \cup \{\mathsf{mux}[\mathtt{i}] |\ \mathtt{i} \geq 0\} \cup \{\mathsf{demux}[\mathtt{i}] |\ \mathtt{i} \geq 0\} \cup \{\mathsf{gc}[\mathtt{i}] |\ \mathtt{i} \geq 0\}$; the integer $\mathtt{i}$ in (de)muxes and gc's is the *threshold* of the link. Allowed links and nodes are drawn in Figure 3. The source nodes of a link are its *premises*; the target nodes are the *conclusions*. Premises and conclusions are assumed to be distinguishable (i.e., we will have left/right premises, $\mathtt{i}$-th conclusion and so on), with the exception of $\bullet$-links. Those nodes that are not premises of any link are the *net conclusions*; unary (de)muxes are also called *lifts*. Each node is the conclusion of exactly one link and premise of at most one link. For all the links (but $\mathsf{gc}, \mathsf{mux}, \mathsf{demux}$ and weakening) we have the constraint that if a premise/conclusion formula is $\varnothing$ then all the other connected nodes must also be labeled with $\varnothing$. We distinguish the premises (conclusions) of muxes (demuxes) with names (which we call *ports* and are denoted in Figure 3 with $(\mathtt{a}_1 \cdots \mathtt{a}_k)$); moreover, each port is of kind *identity* or *garbage*.

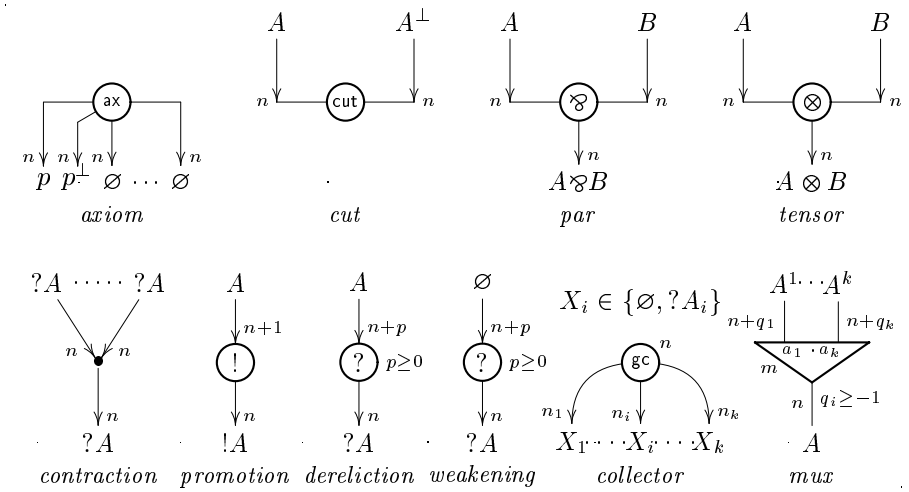


Figure 3. sℓ-structure links

In Figure 3 we have represented a generic mux/demux; more precisely in a mux (demux) the nodes $\mathsf{A}^1, \ldots, \mathsf{A}^k$ are premises (conclusions) and the node A is a conclusion (premise).

With respect to proof nets, sℓ-structures have two additional types of link (mux/demux and gc) and a new constant ($\varnothing$). The $\varnothing$ it is used (at first) for introducing weakening formulas. Indeed, the key technical point of our approach is to avoid nullary premise links to introduce weakening formulas. A $\varnothing$ premise (always connected to an axiom) distinguishes a ?-link used as a weakening from a ?-link used as ?-introduction. During the cut-elimination process, moreover, $\varnothing$ will be used to mark those parts of the proof-structure that have to be discarded (because cut against a weakening).

Muxes (introduced in [Gue96, GMM96]) are responsible for the processes of:

1. reindexing of formulas (that is, the local re-computation of boxes during reduction);
2. local (lazy) duplication;
3. marking of garbage.

The link gc (garbage collector) is designed to collect the garbage—to remove from the net those nodes which have been marked $\varnothing$.

Definition 3.2 (proof ℓ-structure). A *proof ℓ-structure* is an sℓ-structure without (de)muxes and gc links.

3.2 Decoration

It is well known (see [Gir87], pag. 63) that proof nets may be formulated with *several weakenings in the same weakening box.* With this formulation, it is easy

to show (a trivial induction) that by suitable permutations any proof net can be transformed into an equivalent proof net in which each weakening box contains exactly one axiom link as interior. Let $\mathcal{PN}$ be the set of proof nets with such a structure. Since any weakening box contains exactly one axiom link, we may forget the boxes and simply record the weakening formulas with each axiom. Only exponential boxes survive in $\mathcal{PN}$.

We will now show how to associate to each $P \in \mathcal{PN}$ a (unique!) proof ℓ-structure $\mathscr{D}[P]$, the *decoration* of P. The proof ℓ-net $\mathscr{D}[P]$ is obtained by applying the following steps:

1. assign to each node of P a level, corresponding to the number of exponential boxes containing that node;
2. connect each weakening $?A$ to the axiom α belonging to the weakening box of $?A$ by means of a node labeled $\varnothing$;
3. set the level of this $\varnothing$ premise to the number of exponential boxes containing α.

Definition 3.3. A proof ℓ-structure S is a *proof ℓ-net* iff $S = \mathscr{D}[P]$ for some $P \in \mathcal{PN}$.

By using indexes it is possible to "recognize" exponential boxes:

Definition 3.4. Let S be a proof ℓ-structure and let A^k be a premise of a !-link; we call *box* of A^k a sub-hypergraph $bx_S[A^k]$ of S verifying the following properties:

1. $A^k \in bx_S[A^k]$ (A^k is the *principal door* of $bx_S[A^k]$);
2. $bx_S[A^k]$ is a proof ℓ-net;
3. each net conclusion of $bx_S[A^k]$ different from the principal door is a premise, in S, of a ?-link with conclusion at level $j < k$ (such ?-premises are the *secondary doors* of the box);
4. for each $B^j \in S$, if $B^j \in bx_S[A^k]$, then $j \geq k$.

We denote by $BX[S]$ the set of boxes of S. Because of the definition of ℓ-structure, boxes are connected.

4 Reduction

The new information we added to proof nets allows a clear notion of local and asynchronous computation as a fully distributed execution of the standard cut-elimination process (as defined by [Gir87]). The reduction procedure is described by rules of three kinds: logical reduction (β_l), bookkeeping (π), garbage collection ($\sigma_\varnothing$). The proposed approach extends our previous work [GMM96].

4.1 Logical reduction and bookkeeping

The rules β_l, drawn in Figure 4, implement a local version of the usual cut-elimination process (which we indicate as β_s). The only difference with this usual process is when an exponential cut is reduced. In this case, no duplication, reindexing, or erasing of boxes is done. Such operations are only indicated by the introduction of suitable muxes.

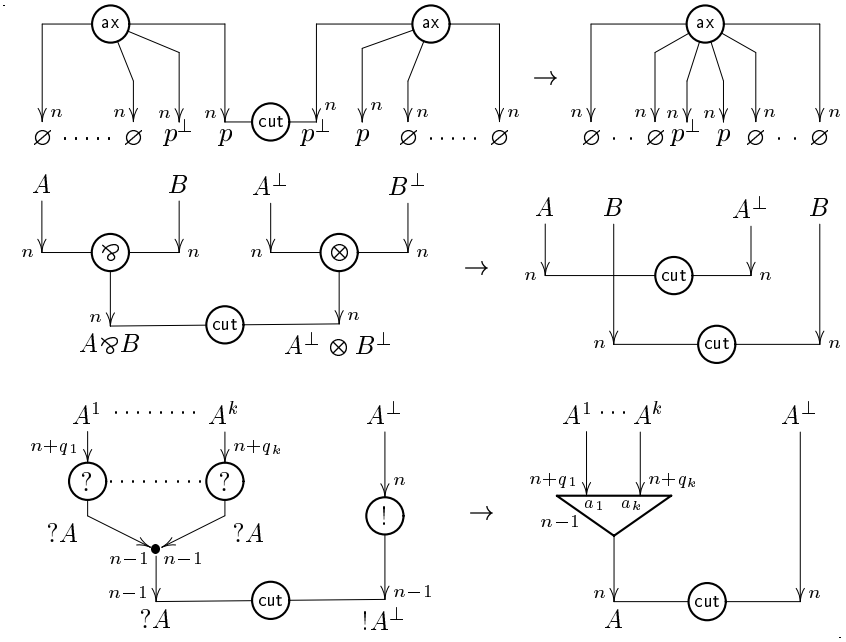


Figure 4. Cut elimination rules: β_l

The bookkeeping rules (π) are responsible for this incremental duplication, reindexing, or erasing. Muxes will travel along the net, duplicating (according to the mux arity) and reindexing all the links they find (Figure 5 shows a generic reduction, where $\star$ stands for any link, but mux).

When one mux encounters another mux, there are two cases, see Figure 6. In the first one, the two muxes had been generated by the same exponential cut (a fact testified by the same threshold for the two muxes): each one has exhausted its job and they disappear. In the second, the two muxes had been generated by two cuts (and hence they have different thresholds): each mux duplicate the other one.

The propagation of muxes ends when either they annihilate by the rule just seen in Figure 5, or when they reach an auxiliary door of the box on which they

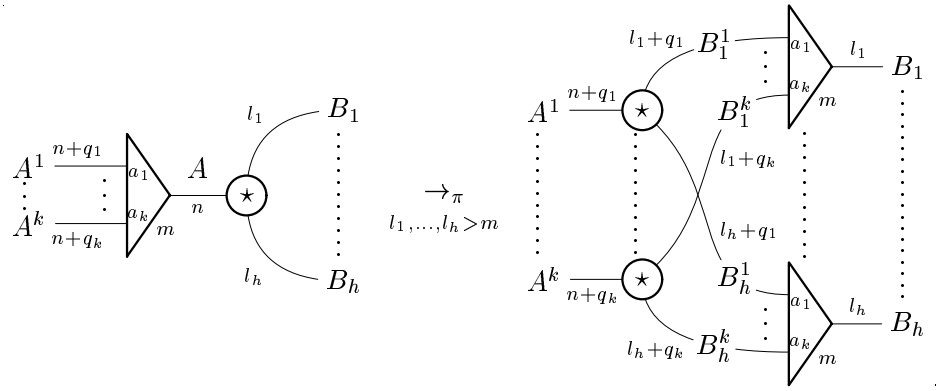


Figure 5. Mux propagation

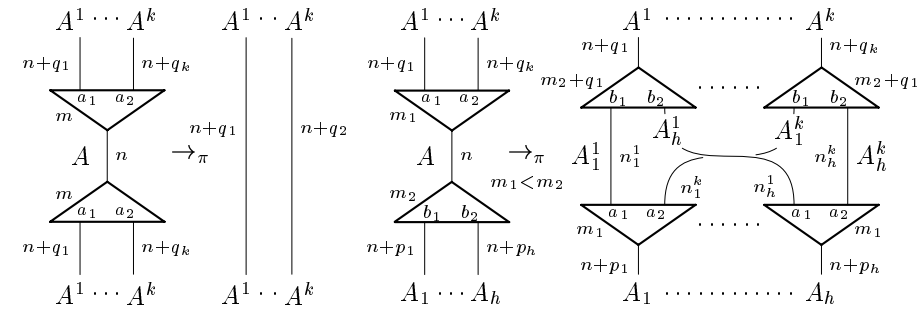


Figure 6. Mux interaction

operate. In this case the mux disappears, absorbed by the corresponding ? links, see Figure 7.

It remains to be discussed how this bookkeeping handles the marking of boxes to be erased (because cut against weakenings). Here come to the stage the *ports* of the muxes (we recall that any mux premise has a name and a kind (*identity* or *garbage*); we call *port* this information). Let us consider the last rule of Figure 4, the creation of a demux.

A port $\mathtt{a}_i$ is of kind *garbage* iff the formula $\mathtt{A}^i$ is a $\varnothing$ premise of a weakening link (which means that $\mathtt{A}^i$ is $\varnothing$ and is the conclusion of an axiom link); otherwise $\mathtt{a}_i$ will be an identity port.

Let us consider now Figure 5 and let $\mathtt{A}^i = \varnothing$ and let the kind of $\mathtt{a}_i$ be *garbage*. We stipulate that in this case, after the reduction, all the $\mathtt{B}^i_1, \ldots, \mathtt{B}^i_h$ formulas are $\varnothing$. Otherwise (i.e., if $\mathtt{a}_i$ is an *identity* port) $\mathtt{B}^i_1, \ldots, \mathtt{B}^i_h$ are syntactically equal to $\mathtt{B}_1, \ldots, \mathtt{B}_h$ respectively. The same convention applies to all the other rules involving a mux propagation (Figures 6 and 10).

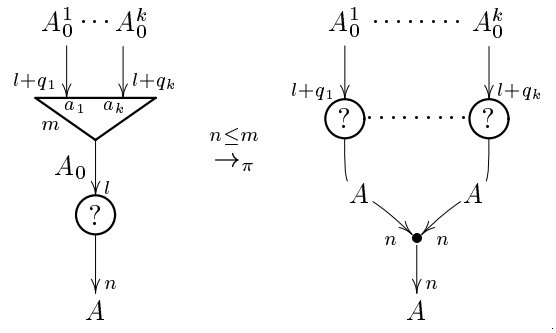


Figure 7. Mux absorption: ? link case

Remark 4.1. A simple rule inspection shows that each node connected to a garbage port must be a $\varnothing$.

4.2 Garbage Collection

We have seen that, during its propagation, a mux with a garbage port marks the net it visits, converting all the formulas into $\varnothing$. This process leaves garbage, whose collection is responsibility of the gc link, by means of the $\sigma_\varnothing$ rules, which correspond to the parsing of garbage subnets (see [GM96] for the problem of parsing proof nets). The collection starts from any axiom whose conclusions are all $\varnothing$: the axiom is transformed into a gc link, see Figure 8. Subsequently (see Figure 9), the gc link keeps "eating" the dummy marked net, collapsing it into a single gc link and collecting all the secondary doors of the box to be deleted (see the last rule of Figure 9).

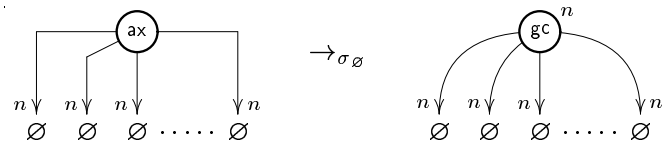


Figure 8. Sweep rules: start collecting

Note that gc does not delete muxes. The interaction between muxes and gc's is regulated by the rule in Figure 10.

The collection ends when the box to be erased is collapsed into a unique gc node. The conclusions of this node are all the secondary doors of the box, plus a single $\varnothing$ conclusion, cut against another $\varnothing$ (coming from the original

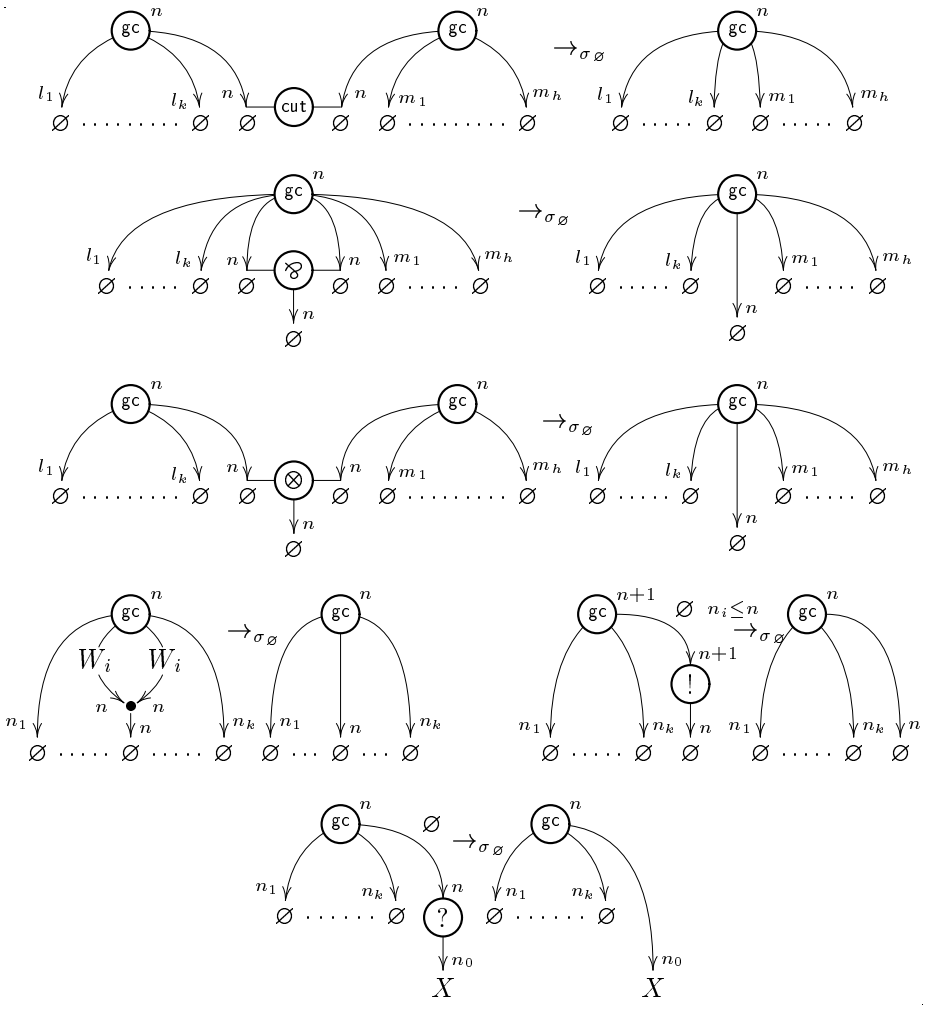


Figure 9. Sweep rules

weakening). This configuration and the corresponding rewriting are depicted in Figure 11. Observe that the gc link disappears and that the secondary doors of the collected box are transformed into weakenings.

5 Results

We may summarize the main result of the paper as: *"cut elimination (with garbage collection) in proof nets may be performed in a completely local and asynchronous way"* .

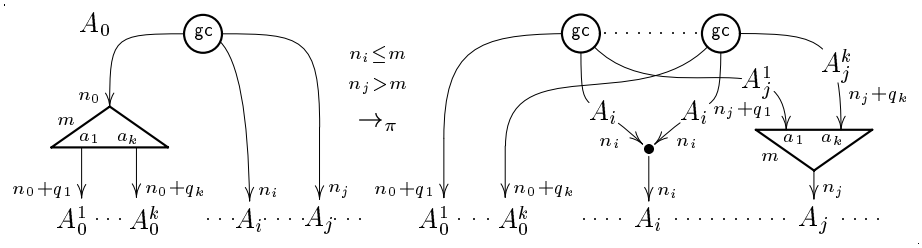


Figure 10. Mux absorption: gc link case

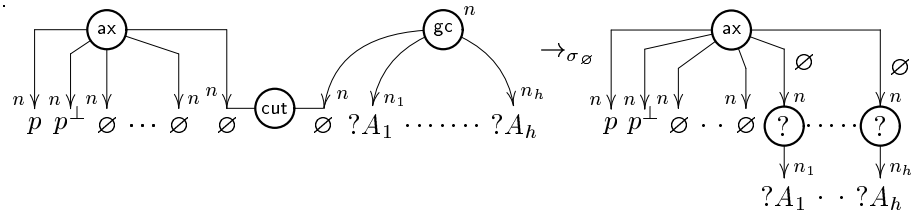


Figure 11. Sweep rules: end collecting

The proofs of theorems stated below are long and technically complex and cannot be inserted here. On the other hand, the techniques used in proofs are not essential for a full understanding of the proposed approach.

Figure 12 lists the different reduction relations used in the statements. The reduction β_s (standard reduction) refers to the usual cut elimination in proof nets, that is, global box duplication, reindexing, or erasing (formulated of course using levels and Definition 3.4). In diagram construction, a dotted arrow means the existence of the corresponding reduction, as usual.

arrow	reduction
$\twoheadrightarrow$	β_s (standard)
$\longrightarrow\!\!\triangleright$	β_l (local)
$\Longrightarrow$	$\pi + \sigma_\varnothing$
$\longrightarrow$	$\pi + \sigma_\varnothing + \beta_l$

Figure 12. One-step reductions

The relevant sℓ-structures we are interested in are the ones arising along the reduction of a proof ℓ-net.

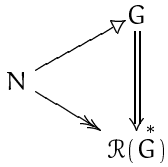
Definition 5.1 (proof sℓ-net). An sℓ-structure G is a *proof sℓ-net* if there exists a reduction $\mathsf{N} \longrightarrow^{*} \mathsf{G}$, for some proof ℓ-net N.

Theorem 5.2 (existence and uniqueness of readback). *For any proof sℓ-net G, we have that:*

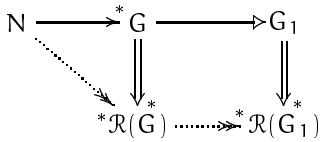
1. *The $\pi + \sigma_{\varnothing}$ rules are strongly normalizing and confluent on G.*
2. *The normal form of G is a proof ℓ-net.*

Definition 5.3 (readback). The *readback* $\mathcal{R}(\mathsf{G})$ of an sℓ-net G is the $\pi + \sigma_{\varnothing}$ normal form of G.

Theorem 5.4 (standard strategy). *For any β redex of a proof ℓ-net N we have that:*



Theorem 5.5 (coherence). *For any proof sℓ-net G and any proof ℓ-net N for which $\mathsf{N} \longrightarrow^{*} \mathsf{G}$, we have that:*



Remark 5.6. The previous result differs from the analogous of [GMM96] in the fact that we cannot ensure that to a β_{l} contraction of G corresponds a non-empty β_{s} reduction of $\mathcal{R}(\mathsf{G})$. In fact, G may not be garbage free, since it may contain part of a box to be erased. Hence, the contraction of a cut in such a part has no correspondence in $\mathcal{R}(\mathsf{G})$. As a consequence, the proof of the strong normalization property becomes more involved.

Theorem 5.7 (strong normalization). *There is no infinite reduction of a proof sℓ-net.*

Theorem 5.8 (confluence). *For any proof sℓ-net G such that $\mathsf{G} \longrightarrow^{*} \mathsf{G}_1$ and $\mathsf{G} \longrightarrow^{*} \mathsf{G}_1$, there exists an s$\ell$-net G_0 such that $\mathsf{G}_1 \longrightarrow^{*} \mathsf{G}_0$ and $\mathsf{G}_2 \longrightarrow^{*} \mathsf{G}_0$.*

Corollary 5.9 (unique normal form). *The $\beta_{\mathsf{l}} + \pi + \sigma_{\varnothing}$ normal form of a proof ℓ-net N is unique and coincides with its β_{s} normal form.*

6 Conclusions

6.1 Discussion

We have presented a distributed and local graph-rewriting algorithm for MELL proof nets. The relations of this work with the ideas and techniques developed for the optimal reduction of interaction nets have been discussed in the Introduction. One may wonder why there is no statement on this subject in the paper. The crucial reason is that, in presence of weakening, the very notion of optimal reduction for proof nets is highly an open question.

It is clear that the mux propagation rule, when applied with a duplicating mux facing the premise of a logical node, would duplicate any redex in its scope, thus destroying any optimality. It is easy to split the propagation rule in two rules, one for when the mux faces a premise (call it the *dup* rule) and one for when the mux faces the conclusion (the *odup* rule). Let now π_{opt} be the subset of π including *odup* but not *dup* (except for axioms, cuts, and gc).

Theorem 6.1. *Let* G *be a proof* sℓ*-net and* N *be its* $\beta+\pi+\sigma$ *normal form. Let* G' *be a* $\beta+\pi_{opt}+\sigma$ *normal form of* G*, then* $\mathcal{R}(\mathsf{G}')=\mathsf{N}$.

Therefore, normalization of proof sℓ-nets may be performed in two distinct steps: first "optimal" reduction ($\beta+\pi_{opt}+\sigma$), then read-back reduction. However, this theorem only warrants that no duplication of redexes is done, but nothing is said about the *need* of such redexes (that is, whether these redexes belong to a part of the net which will not be erased). We are missing, in other words, an effective strategy ensuring the reduction of only needed cuts (like the left-most-outermost in the case of λ-calculus). Let us note, incidentally, that even for sharing graphs for λ-calculus, the meaning of an optimal reduction in presence of weakening is still a non completely understood subject.

6.2 Further Work

All the results of this paper hold for full MELL, i.e., MELL with the two constant $\bot, 1$. Such an extension will be developed in a forthcoming full paper. The basic ideas are the following. The constant 1 is introduced by means of a new axiom link, treated as all the other axioms. The $\bot$ constant is treated like weakening formulas, namely: (1) there is a *bottom* link with a $\varnothing$ premise (connected to an axiom) and the $\bot$ constant as conclusion; (2) the concept of box is extended in order to allow $\bot$ as secondary door (such an extension do not increase the expressive power of the logic, [GM96]). All the results stated here extend rather simply to full MELL.

We plan to apply the proposed approach to the case of functional languages (pure and typed λ–calculi) both from a theoretical and practical (implementative) point of view [AG97].

It would be interesting to develop a semantics of the reductions of this papers, along the lines of the geometry of interaction [Gir89], or of the consistent path semantics for sharing graphs of [ADLR94]. For the weakening-free case, see [Gue96].

References

ADLR94. Andrea Asperti, Vincent Danos, Cosimo Laneve, and Laurent Regnier. Paths in the lambda-calculus: three years of communications without understanding. In *Proceedings of 9th Annual Symposium on Logic in Computer Science*, pages 426–436, Paris, France, July 1994. IEEE.

AG97. Andrea Asperti and Stefano Guerrini. *The Optimal Implementation of Functional Programming Languages*. Cambridge University Press, 1997. To appear.

AL96. Andrea Asperti and Cosimo Laneve. Interaction system II: The practice of optimal reductions. *Theoretical Computer Science*, 159(2):191–244, 3 June 1996.

Asp95. Andrea Asperti. Linear logic, comonads and optimal reductions. *Fundamenta infomaticae*, 22:3–22, 1995.

Ban95. Richard Banach. Sequent reconstruction in LLM—A sweepline proof. *Annals of Pure and Applied Logic*, 73:277–295, 1995.

GAL92. Georges Gonthier, Martín Abadi, and Jean-Jacques Lévy. Linear logic without boxes. In *Proceedings of 7th Annual Symposium on Logic in Computer Science*, pages 223–234, Santa Cruz, CA, June 1992. IEEE.

Gir87. Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

Gir89. Jean-Yves Girard. Towards a geometry of interaction. Number 92 in Contemporary Mathematics, pages 69–108. AMS, 1989.

GM96. Stefano Guerrini and Andrea Masini. Parsing MELL proof nets. Technical report, Dipartimento di Informatica, Pisa, 1996.

GMM96. Stefano Guerrini, Simone Martini, and Andrea Masini. Coherence for sharing proof-nets. In Harald Ganzinger, editor, *RTA '96*, number 1103 in LNCS, pages 215–229, New Brunswick, NJ, July 1996. Springer-Verlag.

Gue96. Stefano Guerrini. *Theoretical and Practical Issues of Optimal Implementations of Functional Languages*. Phd thesis, Dipartimento di Informatica, Pisa, 1996. TD-3/96.

Lam90. John Lamping. An algorithm for optimal lambda calculus reduction. In *Proceedings of Seventeenth Annual ACM Symposyum on Principles of Programming Languages*, pages 16–30, San Francisco, California, January 1990. ACM.

Lév78. Jean-Jacques Lévy. *Réductions Correctes et Optimales dans le lambda-calcul*. PhD Thesis, Université Paris VII, 1978.

MM95. S. Martini and A. Masini. On the fine structure of the exponential rule. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 197–210. Cambridge University Press, 1995. Proceedings of the Workshop on Linear Logic, Ithaca, New York, June 1993.