Proof nets, Garbage, and Computations*

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

[1] Dipartimento di Informatica, Università di Pisa,
Corso Italia, 40, I-56125 Pisa – Italy; guerrini,masini@di.unipi.it.
[2] Dipartimento di Matematica e Informatica, Università di Udine,
Via delle Scienze, 206, I-33100 Udine – Italy; martini@dimi.uniud.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 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

* Partially supported by: HCM Project CHRX-CT93-0046 and CNR GNSAGA (Guerrini,Martini); by BRA 8130 LOMAPS (Masini); by MURST 40% "Modelli della computazione e dei linguaggi di programmazione".

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 [AL93]; 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 net 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].

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 and informally introduces 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. Section 6 concludes.

The paper may be read as it stands. For the convenience of the interested referee, however, we have included an appendix with a detailed roadmap to the proofs.

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.

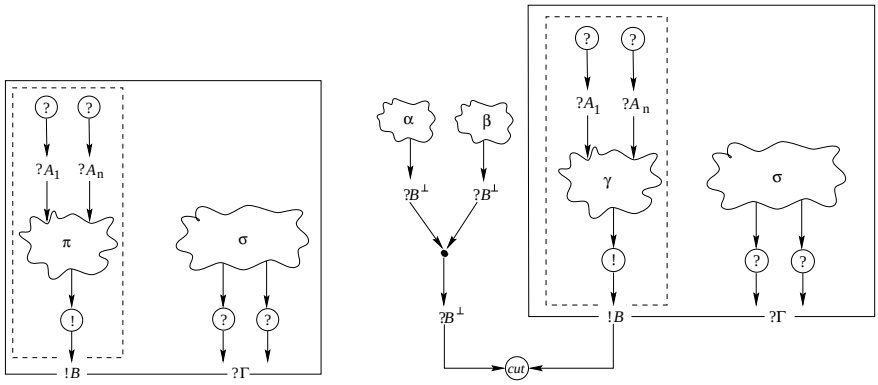


Figure 1. Weak-isles

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, which not only is not the intended reduct of the cut, but neither is a proof net, nor can be made into one by adding boxes. There are two weakening isles, while only one copy of the proof net σ, which cannot validate both.

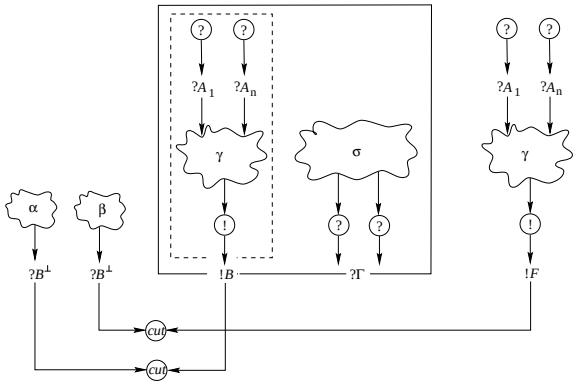


Figure 2. A mistake

To solve the connected components problem we change the definition of the axiom link. Beside the dual atoms $\mathfrak{p}$ and $\mathfrak{p}^\perp$, 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 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, i.e., a box-free representation of (shared) proof structures.

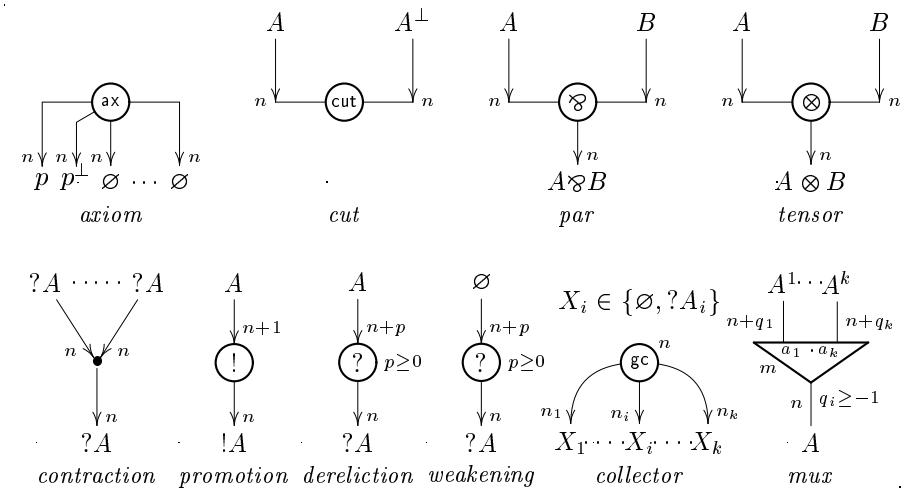


Figure 3. ℓ-structure links

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 $\{\mathsf{cut}, \mathsf{ax}, \parr, \otimes, !, ?, \bullet\} \cup \{\mathsf{mux[i]} \mid \mathsf{i} \geq$

$0\} \cup \{\mathsf{demux}[i] |\ i \geq 0\} \cup \{\mathsf{gc}[i] |\ i \geq 0\}$; the integer 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, 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*. For all the links (but gc, mux, demux and weakening) we have the constraint that if a premise/conclusion formula is $\varnothing$ then all the other connected nodes must also be labelled with $\varnothing$. We distinguish the premises (conclusions) of muxes (demuxes) with names (which we call *ports* and are denoted in Figure 3 with $(a_1 \cdots a_k)$); moreover, each port is of kind *identity* or *garbage*.

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

Definition 3.2 (ℓ-structure). An ℓ-structure is an sℓ-structure in which each node is the conclusion of exactly one link and premise of at most one link.

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 for the first time 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 3.3 (proof ℓ-structure). A *proof ℓ-structure* is an sℓ-structure without (de)muxes and gc links.

3.2 Decoration

It is well know (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 $S \in \mathcal{PN}$ a (unique!) proof ℓ-structure $\mathscr{D}[S]$, called *decoration* of S. The proof ℓ-net $\mathscr{D}[S]$ is obtained by applying the following steps:

1. assign to each node of S 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 labelled $\varnothing$;
3. set the level of this $\varnothing$ premise to the number of exponential boxes containing α.

Definition 3.4. A proof ℓ-structure S is a *proof ℓ-net* iff $S = \mathscr{D}[Q]$ for some $Q \in \mathcal{PN}$.

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

Definition 3.5. Let Π be a proof ℓ-structure and let A^k be a premise of a !-link; we call *box* of A^k a sub-hypergraph $\mathfrak{box}_\Pi[A^k]$ of Π verifying the following properties:

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

We denote by $\mathfrak{BX}[\Pi]$ the set of boxes of Π. Because of the definition of ℓ-structure, boxes are connected.

4 The reduction system

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 significatively extends our previous work [GMM96].

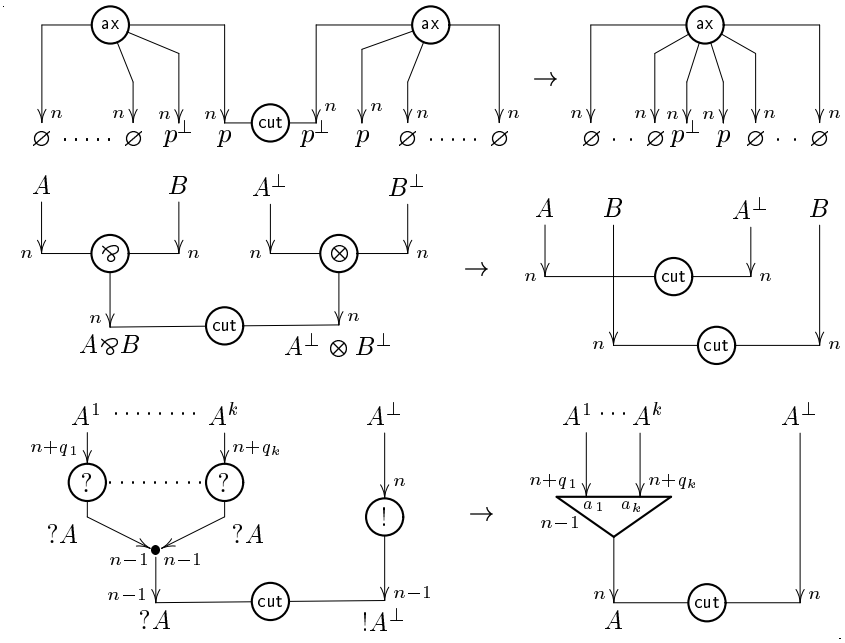


Figure 4. Cut elimination rules: β_l

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.

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

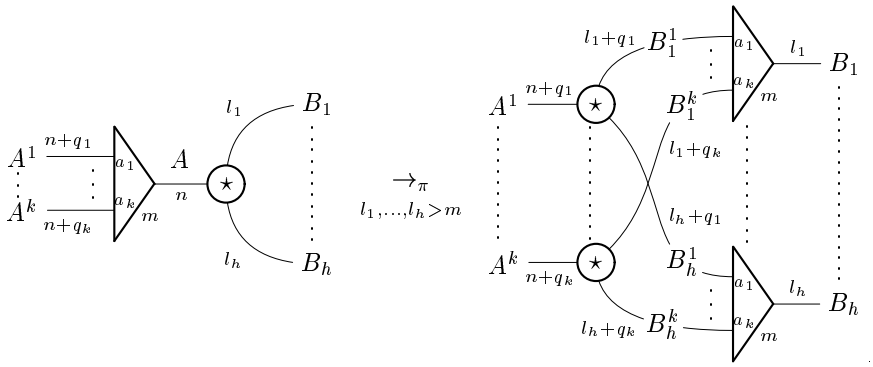


Figure 5. Mux propagation

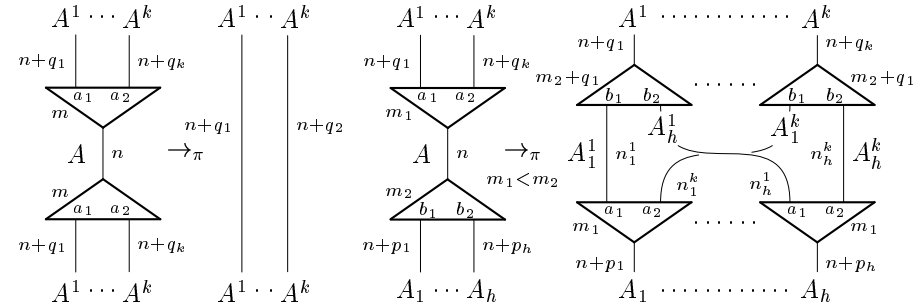


Figure 6. Mux interaction

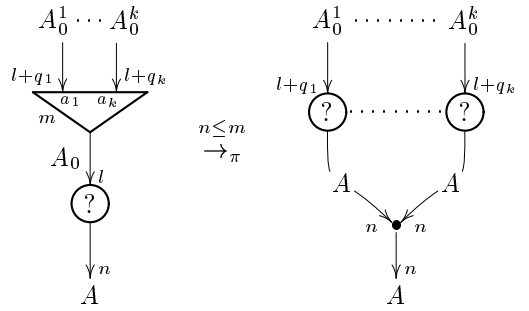


Figure 7. Mux absorption: ? link case

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

Let us consider now Figure 5 and let $A^i = \varnothing$ and let the kind of a_i be *garbage*. We stipulate that in this case, after the reduction, all the $B^i_1, \ldots, B^i_h$ formulas are $\varnothing$. Otherwise (i.e., if a_i is an *identity* port) $B^i_1, \ldots, B^i_h$ are syntactically equal to $B_1, \ldots, B_h$ respectively. The same convention applies to all the other rules involving a mux propagation (Figures 6 and 10).

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).

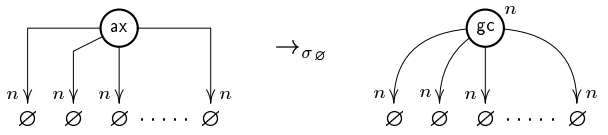


Figure 8. Sweep rules: start collecting

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 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.

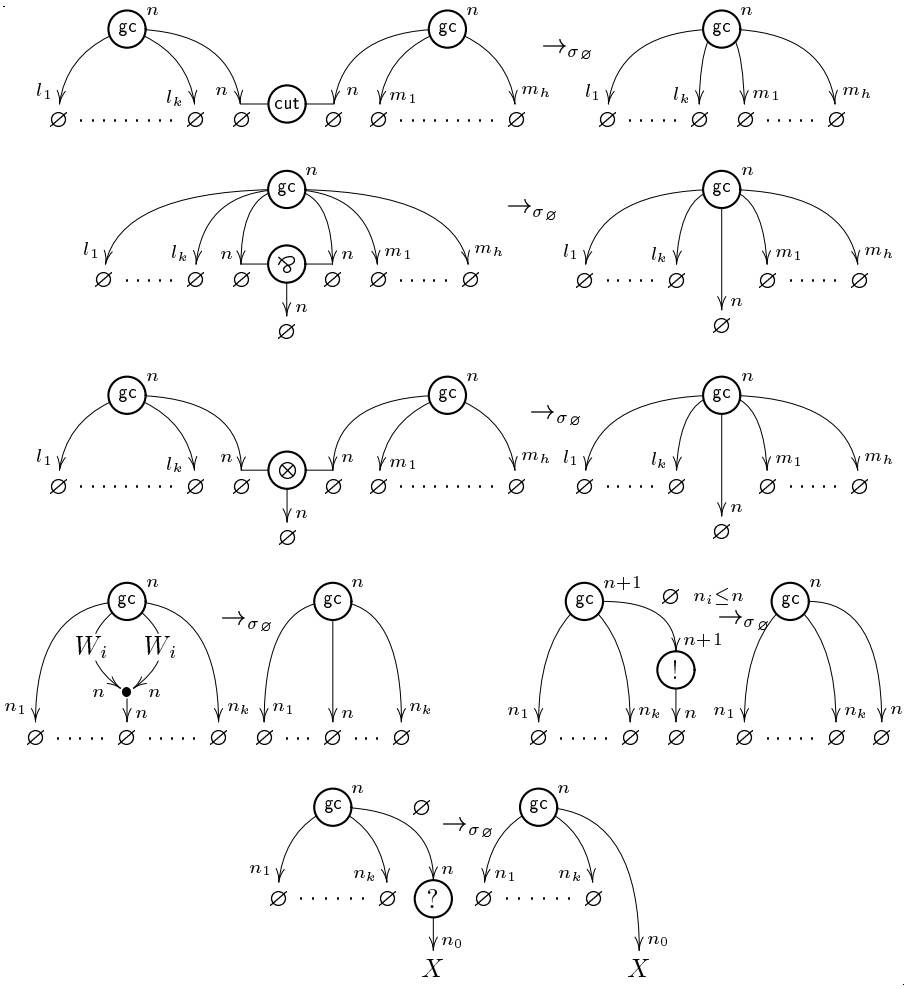


Figure 9. Sweep rules

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"* .

The proofs of theorems stated below are long and technically complex. Space limitations forbid the insertion not only of the full proofs, but also of intelligible sketches of them. On the other hand, the techniques used in proofs are not essential for a full understanding of the proposed approach.

For the sake of the interested referee, we have included an appendix from

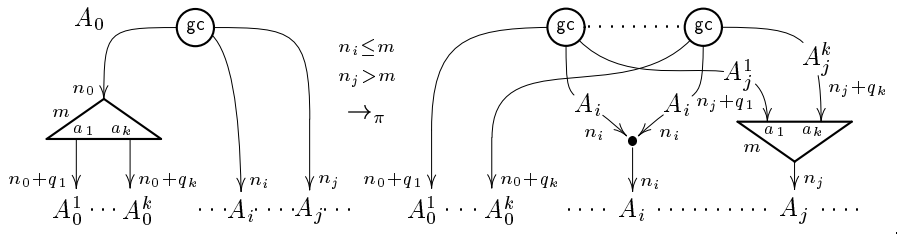


Figure 10. Mux absorption: gc link case

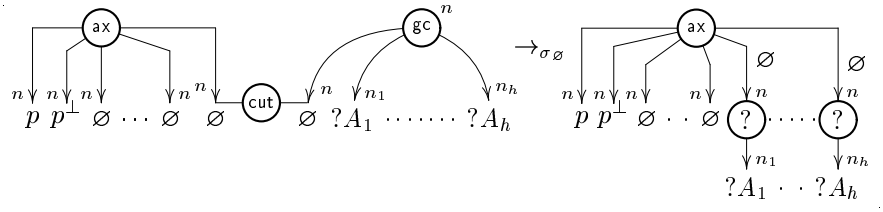


Figure 11. Sweep rules: end collecting

which the reconstruction of full proofs is not difficult.

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 3.5). In diagram construction, we assume the standard convention that a dotted arrow means the existence of the corresponding reduction.

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

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

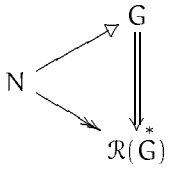
exists a reduction $\mathsf{N} \longrightarrow^* \mathsf{G}$, for some proof ℓ-net N.

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

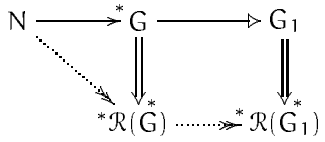
1. *The* $\pi + \sigma_\varnothing$ *rules are strongly normalizing and confluent on* G*.*
2. *The normal form of* G *is a proof* ℓ*-net.*

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_ι 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 *s.t.* $\mathsf{G} \longrightarrow^* \mathsf{G}_1$ *and* $\mathsf{G} \longrightarrow^* \mathsf{G}_1$*, there exist an* s$\ell$*-net* G_0 *s.t.* $\mathsf{G}_1 \longrightarrow^* \mathsf{G}_0$ *and* $\mathsf{G}_2 \longrightarrow^* \mathsf{G}_0$*.*

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

6 Conclusions

All the results obtained here hold for full MELL, i.e., MELL with the two constant $\bot, 1$. Such an extension has not been reported here for space limits and will be developped 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 in a similar way of 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). 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, with a particular emphasis on the problem of optimality.

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.

AL93. Andrea Asperti and Cosimo Laneve. Interaction systems. In *Int. Workshop on Higher Order Algebra, Logic and Term Rewriting*, 1993.

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.

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 *RTA '96*, New Brunswick, NJ, July 1996. To appear.

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.

APPENDIX: Proofs of the Theorems.

A Definitions

We extend $s\ell$-structures to parsing proof $s\ell$-structures by introducing a new link called seq. A seq is a link with k conclusions Γ and no premises. There is no restriction on the shape and number of the conclusions of seq links. We will use seq links as a sort of generalized axioms, that is, it will be always true that the sequent $\vdash \Gamma$ is provable in MELL.

Definition A.1 (parsing). Let us denote with $\sigma_\bullet$ the set of rewriting rules derived from the sweep rules $\sigma_\varnothing$ by:

1. Replacing each gc link by a seq link.
2. Relaxing the condition on the type of the formulae involved in the redexes, i.e., the dummy nodes are replaced by ordinary MELL formulae according to the usual constraints established by the types of the links.

We denote with σ the union of $\sigma_\varnothing$ and $\sigma_\bullet$ rules.

For a detailed discussion of parsing see [GM96]. We only stress that:

Theorem A.2. *A proof ℓ-structure* G *with conclusions* Γ *is a proof ℓ-net iff* $G \xrightarrow{*}_{\sigma_\bullet} \mathsf{seq}_\Gamma$ *(where* seq_Γ *denotes a structure formed of a single* seq *link with conclusions* Γ*).*

The extension of the parsing rules $\sigma_\bullet$ corresponds to the extension of the symbols for rewriting in Figure 13.

arrow	reduction
$\twoheadrightarrow$	β_s (standard)
$\multimap$	β_l (local)
$\Longrightarrow$	$\pi + \sigma_\varnothing$
$\Longrightarrow_\bullet$	$\pi + \sigma$
$\rightarrow$	$\pi + \sigma_\varnothing + \beta_l$
$\rightarrow_\bullet$	$\pi + \sigma + \beta_l$

Figure 13. Legend

For the sake of the proofs we add to the π-rules the mux permutation rule of Figure 14.

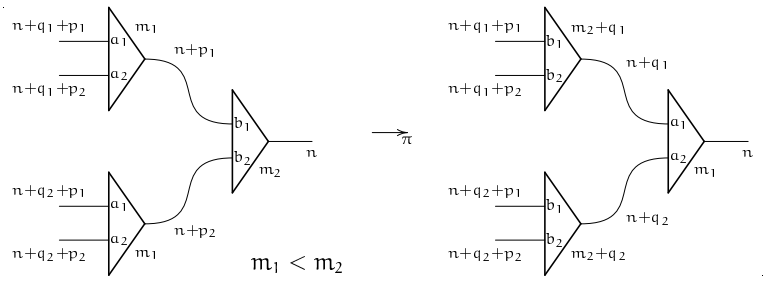


Figure 14. Mux permutation rule

A mux is deadlocked when either: *i)* its principal port (its vertex) is connected to a conclusion of the net; or *ii)* its principal port is connected to any port of another link, but a secondary port of a mux, and does not form a π redex. In particular, a mux with threshold n is deadlocked when faces to a non complementary mux with threshold n and when it reaches the premise of an ! link whose conclusion is at level n. Summarizing, a mux is deadlocked only when we are sure that it cannot move anymore. According to this, a mux is not deadlocked when it follows another mux, even if they do not permute.

An $s\ell$-structure is *deadlock free* when none of its reducts contains deadlocked muxes.

Definition A.3. Let G be a parsing $s\ell$-structure. We define the predicates:

1. $\mathsf{CORR}(G)$ iff $G \xrightarrow[\bullet]{*} \mathsf{seq}_\Gamma$.
2. $\mathsf{LC}_\pi(G)$ iff G is locally confluent w.r.t. π (modulo the mux permutation of Figure 14).
3. $\mathsf{DF}_\pi(G)$ iff G is deadlock free.
4. $\mathsf{SN}_\alpha(G)$ iff G is strongly normalizing w.r.t. a set α of rewriting rules.

Furthermore we use $\mathcal{R}_\pi(G)$ ($\mathcal{R}(G)$) to denote a normal form (if any) of G w.r.t. π ($\pi + \sigma_\varnothing$).

A first relevant property to note is that deadlock freeness is invariant under mux permutation.

Lemma A.4. *If* G_1 *and* G_2 *are equal modulo mux permutation, then* $\mathsf{DF}_\pi(G_1)$ *iff* $\mathsf{DF}_\pi(G_2)$.

Proof. By case analysis.

An immediate consequence of the previous definitions and of Lemma A.4 is the following corollary, which is fundamental to prove the uniqueness of the readback (Theorem 5.2).

Corollary A.5. *If* $\mathsf{DF}_\pi(G)$ *and* $\mathsf{SN}_\pi(G)$ *and* $\mathsf{LC}_\pi(G)$, *then* $\mathcal{R}_\pi(G)$ *exists unique and is mux free.*

B Results

Following the mainstream of [Gue96, GMM96] the proofs of the results split in two main parts:

1. The definition and analysis of an unshared case.
2. The lifting to the shared case of the results proved for the unshared one via a simulation lemma stating that each shared reduction induces a corresponding unshared one.

To study the unshared case we introduce unshared structures, $\mathfrak{u}\ell$-structures for short (see Definition B.1). In $\mathfrak{u}\ell$-structures boxes are still present and used to define the unshared beta reduction $\beta_{\mathfrak{u}}$, in which, even if box reindexing is perfomed by a unary mux called *lift*, box duplication is done in a global way.

Anyway, to gain in readability, we give here the *contraction free* case only. In this way, we can in fact avoid to introduce global duplication of boxes, since for contraction free $\mathfrak{u}\ell$-structures $\beta_{\mathfrak{l}}$ and $\beta_{\mathfrak{u}}$ coincide. Nevertheless, we can present all the relevant material useful for a full understanding of the proof techniques without the technical overhead required by the complete analysis of the unshared case. The reader interested in all the details should apply to the present case the techniques developed in [Gue96, GMM96].

Definition B.1. A proof $\mathfrak{u}\ell$*-structure* is a proof $\mathfrak{s}\ell$-structure in which all the muxes are *lifts*, that is, all the muxes are unary.

In all this section G will be a $\mathfrak{u}\ell$-structure.

Lemma B.2. $\mathsf{DF}_\pi(\mathsf{G})$ *implies* $\mathsf{LC}_\pi(\mathsf{G})$.

Proof. See [Gue96, p. 52].

The previous lemma allows to strengthen Corollary A.5.

Corollary B.3. *If* $\mathsf{DF}_\pi(\mathsf{G})$ *and* $\mathsf{SN}_\pi(\mathsf{G})$, *then* $\mathcal{R}_\pi(\mathsf{G})$ *exists unique and is mux free.*

Lemma B.4. *Let* $\mathsf{G} \longrightarrow_\pi \mathsf{G}'$. $\mathsf{DF}_\pi(\mathsf{G}')$ *implies* $\mathsf{DF}_\pi(\mathsf{G})$

Proof. By inspection of the π rules.

Lemma B.5. *For each reduction sequence* $\rho : \mathsf{G} \Longrightarrow^*_\bullet \mathsf{G}'$ *there is a corresponding reduction sequence* $\rho' : \mathsf{G} \longrightarrow^*_\pi \mathsf{G}_1 \longrightarrow^*_{\sigma_\varnothing} \mathsf{G}_2 \longrightarrow^*_{\sigma_\bullet} \mathsf{G}'$; *where, if in* ρ *there are not* $\sigma_\bullet$ *rules, then* $\rho' : \mathsf{G} \longrightarrow^*_\pi \mathsf{G}_1 \longrightarrow^*_{\sigma_\varnothing} \mathsf{G}'$.

Proof. By case analysis. Studying the commutations between $\pi, \sigma_\varnothing$, and σ rules.

The following corollaries are consequences of the previous lemmas:

Corollary B.6. $\mathsf{CORR}(G)$ *implies* $G \xrightarrow{*}_{\pi} \mathcal{R}_\pi(G) \xrightarrow{*}_{\sigma_@} \mathcal{R}(G) \xrightarrow{*}_{\sigma_\bullet} \mathsf{seq}_\Gamma$.

Corollary B.7. $\mathsf{CORR}(G)$ *implies* $\mathsf{DF}_\pi(G)$.

Corollary B.8. *If* G *is lift free and* $\mathsf{CORR}(G)$ *holds, then* $G \xrightarrow{*}_{\sigma_@} \mathcal{R}(G) \xrightarrow{*}_{\sigma_\bullet} \mathsf{seq}_\Gamma$.

Lemma B.9. $\mathsf{CORR}(G)$ *implies* $\mathsf{SN}_{\pi+\sigma_@}(G)$.

Proof. Let $\mathcal{W}$ be the monoid obtained quotienting the free monoid built on pairs $\langle m, q\rangle$, where $m \geq 0, q \geq -1$ are integers, w.r.t. the equation

$$\langle m_2, q_2\rangle \circ \langle m_1, q_1\rangle = \langle m_1, q_1\rangle \circ \langle m_2 + q_2, q_2\rangle$$

for $m_1 < m_2$. Since, in this way we equate sequences with the same length, let us define $\#\alpha = k$ when $\alpha = \langle m_1, q_1\rangle \circ \cdots \circ \langle m_k, q_k\rangle$.

Let be given a parsing $s\ell$-structure G (i.e., G may contain seq links too). A $\mathcal{W}$-decoration of G is a weight assignment to G which associates to each formula of G an element of $\mathcal{W}$ s.t., between the weights α_1, α_2 assigned to a pair of formulae which are premises or conclusions of a same link ν:

1. If ν is an axiom, par, tensor, contraction, or of-course link, then $\alpha_1 = \alpha_2$.
2. If ν is a why-not, a seq, or a gc link, let n_1 and n_2, with $n_1 \leq n_2$, be the levels of the formulae corresponding to the two weights; then $\alpha_2 = \alpha_1\alpha_0$, with $\alpha_0 \in \mathcal{W}[0, n_1]$ and $\alpha_1 \in \mathcal{W}[n_1, \omega]$, where

$$\mathcal{W}[n, m] = \{\langle m_1, q_1\rangle \circ \cdots \circ \langle m_k, q_k\rangle | n \leq m_i < m + \sum_{j<i} q_i, i = 1, \ldots, k\}$$

3. If ν is a lift with threshold m and offset q, let α_2 be the weight of the formula at the vertex (at the principal port) of ν; then $\alpha_2 = \langle m, q\rangle \circ \alpha_1$.

Let $\mathcal{D}$ be a $\mathcal{W}$-decoration for G, we associate to the graph a pair $|G|_{\mathcal{D}} = \langle h_0, h_1\rangle$, where:

1. $h_0 = \Sigma_\nu \#\alpha_\nu$, for ν ranging over the set of the links which are not lifts; in which α_ν is the weight assigned by $\mathcal{D}$ to the formula near to ν whose level is maximum;
2. $h_1 = \Sigma_\nu \#\alpha_\nu$, for ν ranging over the set of the lifts of G; in which α_ν is the weight assigned by $\mathcal{D}$ to the formula near the vertex of ν.

The previous pairs are lexicographically ordered, that is:

$$\langle h_0, h_1\rangle < \langle h_0', h_1'\rangle \text{ if } h_0 < h_0'$$
$$\langle h_0, h_1\rangle < \langle h_0', h_1'\rangle \text{ if } h_1 < h_1'$$

Claim. Let $G \Longrightarrow_\bullet G'$. We have that:

1. G has a $\mathcal{W}$-decoration iff G' has a $\mathcal{W}$-decoration.

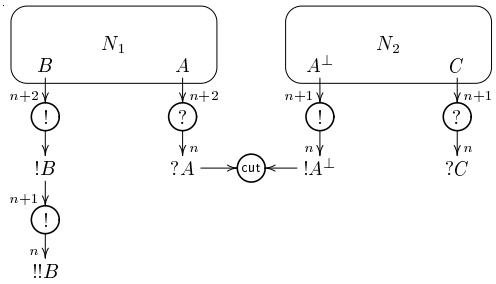
2. If $\mathcal{D}$ is a $\mathcal{W}$-decoration of G, then there exists a $\mathcal{W}$-decoration of G' for which $|\mathsf{G}'|_{\mathcal{D}'} < |\mathsf{G}|_{\mathcal{D}}$.

We omit any detailed description of the proof of the claim, since it just proceed by case analysis.

To conclude, let us note that, by the second item of the claim, an $\mathfrak{u}\ell$-structure with a $\mathcal{W}$-decoration cannot have an infinite $\pi + \sigma_{\varnothing}$ reduction. Hence, since by the first item of the claim we can infer that G has a $\mathcal{W}$-decoration when $\mathsf{CORR}(\mathsf{G})$ holds, we get the statement of the lemma.

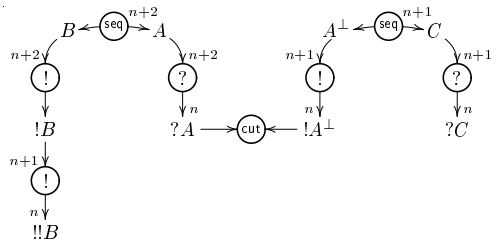
Lemma B.10. *If* $\mathsf{G} \longrightarrow\!\!\!\!\triangleright\, \mathsf{G}'$, *then* $\mathsf{CORR}(\mathsf{G})$ *implies* $\mathsf{CORR}(\mathsf{G}')$.

Proof. We just show the sketch of the proof by an example in the relevant case of an exponential cut. For this purpose, let us assume G composed of two substructures N_1 and N_2 (note that they may contain lifts too) connected by an exponential cut as in the next picture:

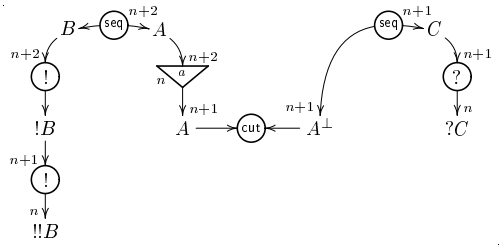


Because of the sequence of two ! links below B, the net N_1 is enclosed into two boxes, while the subnet N_2 is enclosed in just one box.

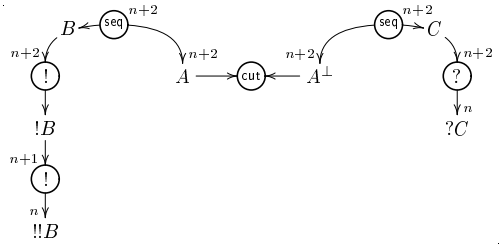
Since $\mathsf{CORR}(\mathsf{G})$ holds, there is a parsing reduction starting from G ending with the following net:



We can now contract the exponential cut by a $\beta_{\mathfrak{u}}$ rule which introduces a lift with offset 1 and threshold $\mathfrak{n}$.



The lift can then propagate through the cut and seq links annihilating at the ? link.



The result, is the net G'' drawn in the previous picture which contracts in few steps to a seq link.

Summarizing, starting from G we have a reduction sequence $\rho = \delta r \gamma$ ending with G'', where: δ is a parsing reduction contracting N_1 and N_2 to a couple of seq links; r is the β_u contraction of the exponential cut; and γ is the propagation of the lift introduced by r.

The reduction ρ can be rearranged executing the β_u contraction at first, that is, obtaining a reduction $\rho' = r\delta'\gamma : G \longrightarrow^*_\bullet G''$. From which we conclude $\mathsf{CORR}(G')$, if $r : G \longrightarrow\!\!\triangleright\, G'$.

Corollary B.11. *For any* $u\ell$*-net* G, $\mathsf{CORR}(G)$ *holds.*

Proof of Theorem 5.2 (existence and uniqueness of readback)

Since by Corollary B.11 we have that $\mathsf{CORR}(G)$ holds, then also $\mathsf{SN}_{\pi+\sigma_\varnothing}(G)$ (by Lemma B.9) and $\mathsf{DF}_\pi(G)$ (by Corollary B.7) hold. By Corollary B.6, $G \longrightarrow^*_\pi \mathcal{R}_\pi(G) \longrightarrow^*_{\sigma_\varnothing} \mathcal{R}(G) \longrightarrow^*_{\sigma_\bullet} \mathsf{seq}_\Gamma$. Hence, $\mathcal{R}_\pi(G)$ is unique and lift free (by Corollary B.3). The uniqueness of $\mathcal{R}(G)$ is then an easy consequence of the simple observation that the $\sigma_\varnothing$ rules are locally confluent and strong normalizing. The proof is concluded by observing that Corollary B.6 also implies that $\mathcal{R}(G)$ is a proof ℓ-net.

Proof of Theorem 5.5 (coherence)

Before to start, let us note that by Lemma B.10 $\mathsf{CORR}(G)$ holds.

The proof is by induction on the length of $N \longrightarrow^* G$. To sketch it, let us just consider the case seen in the sketch of the proof of Lemma B.10. Because of Lemma B.5, the reduction ρ' in the proof of Lemma B.10 can be rearranged again postponing the σ rules to the end of the reduction.

Hence, let us firstly assume that N_1 and N_2 do not contain lifts. After the postponement we obtain the reduction $r\gamma'\delta'' : G \longrightarrow^*_\bullet G''$, where γ' is the propagation of the lift introduced by r through the whole structure N_2. It is not difficult to see that after $r\gamma'$ the levels of the formulae in N_2 are incremented by 1, and then (let $G' = G_1$) that $\gamma' : G_1 \Longrightarrow^* \mathcal{R}(G_1)$ and $G = \mathcal{R}(G) \twoheadrightarrow \mathcal{R}(G_1)$.

Let us now assume that N_1 and N_2 contain lifts. Because of Corollary B.8 we have that $\delta = \delta_2\delta_1$, for a $\sigma_\bullet$ reduction δ_1 and a $\pi + \sigma_\varnothing$ reduction δ_2 s.t. $\delta_2 : G \Longrightarrow^* \mathcal{R}(G)$. During δ_2 some lifts may cross the exponential cut, but it is worth to note that in any case such lifts have a threshold lower than n. The last observation ensures that δ_2, and then δ, commutes with r. Moreover, by Lemma B.5 we can commute γ' and δ_1 obtaining a reduction $r\delta_2'\gamma''\delta_1' : G \longrightarrow^*_\bullet G''$, s.t. $\delta_2'\gamma'' : G_1 \Longrightarrow^* \mathcal{R}(G_1)$ and $\mathcal{R}(G) \twoheadrightarrow \mathcal{R}(G_1)$.

Dead and alive parts of a $u\ell$-net

Given a $u\ell$-net G (i.e., an unshared $s\ell$-net) there is a natural embedding of $\mathcal{R}_\pi(G)$ into G associating each link of $\mathcal{R}_\pi(G)$ to a corresponding link of G s.t. a pair of links of G connected by a formula A are connected by a path of G crossing lifts only (by the way all the nodes of such a path are occurrences of the formula A). By inspection of the $\sigma_\varnothing$ rules we also see that $\mathcal{R}(G)$ is a subnet of $\mathcal{R}_\pi(G)$. The subnet G_a of G associated to $\mathcal{R}(G)$ via the previous embedding is the *alive part* of G. Its complement G_d is instead the *dead part* of G. Correspondingly, we say *dead* any reduction of G internal of G_d, and *alive* any reduction of G internal to G_a.

It is worth noting that, for any β_u reduction $r : G \underset{\beta_u}{\longrightarrow} G_1$, we have that $\mathcal{R}(G) = \mathcal{R}(G_1)$ if r is dead, and $\mathcal{R}(G) \twoheadrightarrow \mathcal{R}(G_1)$ if r is alive. That is, since the dead part of a $u\ell$-net corresponds to its garbage, any dead β_u reduction is useless and does not change the readback.

Proof of Theorem 5.4 (standard strategy)

It is a corollary of Theorem 5.5.

Lemma B.12. *There is no reduction of a proof $u\ell$-net containing an infinite number of β_u contractions.*

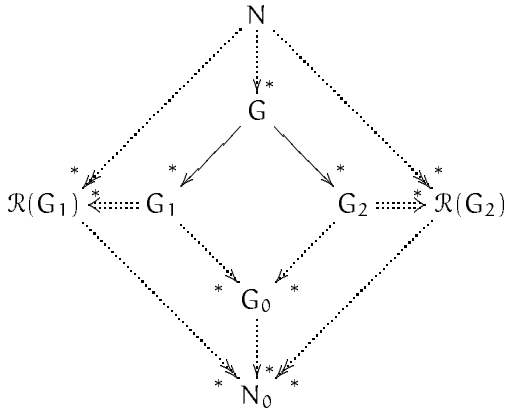
Proof. Let ρ be a reduction sequence containing at least one dead β_u contraction. It is immediate to observe (by Theorem 5.5) that ρ cannot contain an infinite number of alive β_u rules. Let us then suppose that ρ contains a subsequence of alive contractions of lenght k (we are sure that k is finite since $\pi + \sigma$ is SN on $u\ell$-nets). Now, let us consider the first dead β_u contraction d in ρ. There exists an alive β_u contraction a preceding d in ρ s.t. a actives an eraser, and furthermore a is the cause of d to be dead. It is a simple exercise to prove that ρ can be rearranged in a new reduction sequence ρ' where the contraction a is postponed after the execution of d. Since ρ' has a subsequence of alive contractions of lenght $k+1$, we conclude the impossibility of having infinite dead β_u rules contained in ρ.

Proof of Theorem 5.7 (strong normalization)

Let us suppose to have an infinite reduction ρ. By Lemma B.12 we have that there is an infinite subsequence of $\pi+\sigma_\varnothing$ rules, but by the lemmas B.10 and B.9 this is impossible.

Proof of Theorem 5.8 (confluence)

Let us consider the following diagram, in which N, N_0 are proof ℓ-nets:



Its inner diamond corresponds to the statement of the theorem. The commutativity of the upper half of the diagram follows from Theorem 5.5. The commutativity of the external diamond is a consequence of confluence of β_s. To get the remaining arrows, let us note that by Theorem 5.4 we have $G_1 \longrightarrow^* \mathcal{R}(G_1) \longrightarrow^* N_0$ and $G_2 \longrightarrow^* \mathcal{R}(G_2) \longrightarrow^* N_0$. So, a possible choice to complete the diagram is $G_0 = N_0$.

C Generalization

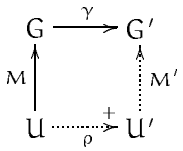
All the previous stated results for proof $u\ell$-structures can be lifted to the general case of proof $s\ell$-structures.

First of all we must define the concept of shared morphism.

Definition C.1. A shared morphism is a surjective graph homomorphism $M : U \to G$, where U is a $u\ell$-structure and G is an $s\ell$-structure, which preserves the link and node labels (i.e., the type of links and the levels of nodes) and the names of the link ports, and is injective when restricted to the conclusions of G.

The key idea is now to have a lemma which allows to simulate unshared reductions by means of usual shared reductions:

Lemma C.2 (simulation). *Let U be a $u\ell$-structure s.t. $DF_\pi(U)$ and let $M : U \to G$ be a shared morphism. For each reduction $\gamma : G \to G'$ there exists a non empty unshared reduction sequence ρ and a shared morphism M' s.t.:*



For the proof of this lemma we refer the reader to [Gue96, GMM96].

It is then a simple exercise to use the simulation lemma to extend all the stated results of the previous section to the shared case.