Under consideration for publication in Math. Struct. in Comp. Science

Linear Logic & Polynomial Time

Damiano Mazza

Institut de Mathématiques de Luminy (UMR 6206)
Campus de Luminy, Case 907 — 13288 Marseille Cedex 9
mazza@iml.univ-mrs.fr
http://iml.univ-mrs.fr/∼mazza

Received 17 January 2006; Revised 6 June 2006

Light and Elementary Linear Logic, the cornerstones at the interface between logic and implicit computational complexity, were originally introduced by Girard as "stand-alone" logical systems with a (somewhat awkward) sequent calculus of their own. The latter has later been reformulated by Danos and Joinet as a proper subsystem of linear logic, whose proofs satisfy a certain structural condition. We extend this approach to polytime computation, finding two solutions: the first one, obtained by a simple extension of Danos&Joinet's condition, closely resembles Asperti's Light Affine Logic and enjoys *polystep strong normalization* (the polynomial bound does not depend on the reduction strategy); the second one, which needs more complex conditions, exactly corresponds to Girard's Light Linear Logic.

1. Introduction

Elementary Linear Logic (**ELL**) and Light Linear Logic (**LLL**), both introduced by Girard (Girard, 1998), are two logical systems characterizing respectively the class of Kalmar elementary functions and the class of deterministic polytime functions. Their syntactical formulation is quite different from that of "traditional" linear logic; in particular, in order to correctly handle additive rules (connectives & and $\oplus$), in the sequent calculus of these systems there appear *blocks* of occurrences of formulas instead of simple occurrences of formulas, thus introducing an "additive layer" below the usual "multiplicative layer".

Unlike virtually any other logical system introduced by Girard, the syntax of "light" logics does not rest upon any semantical background (even though both a truth (Kanovich et al., 2003) and a denotational (Baillot, 2004a) semantics have later been proposed for **LLL**); it is therefore very hard to claim this syntactical formulation to be "definitive" in any sense, and Girard himself still considers the field of complexity-bounded systems "very experimental". Improving the syntax of **ELL** and **LLL** (in particular simplifying it) may thus contribute to a better understanding of how the logical characterization of complexity classes actually works, and at the same time certainly encourages their use and the possible development of their applications.

The first contribution in this direction came from the work of Asperti and Roversi (Asperti and Roversi, 2002), who proposed a quite sharp simplification of Girard's systems by reformulating them in an *affine* setting, i.e., liberalizing weakening and thus completely eliminating additive connectives. Light Affine Logic (**LAL**), and its elementary cousin **EAL**, have ever since been taken as the "standard" interface between implicit computational complexity and linear logic: let us mention for example the work of Coppola&Martini and Baillot on "light" typing algorithms (Coppola and Martini, 2001; Baillot, 2004b), Terui's Light Affine Set Theory (Terui, 2004), and his Light Affine λ-calculus (Terui, 2002).

On the other hand, there is a softer approach to the simplification of "light" logics, namely that of Danos and Joinet (Danos and Joinet, 2003), who redefined **ELL** as the proper subset of linear logic proofs which satisfy the so-called *stratification condition.* Danos and Joinet's approach is interesting for several reasons:

— it takes advantage of existing linear logic tools like proof-nets (instead of having to define new proof systems), getting all of their good properties (confluence, strong-normalization, etc.) almost for free;
— it does not need any intuitionistic restriction;
— it might allow a more integrated view of complexity-bounded logical systems, in that different complexity classes are characterized by different parts of a single system. This could turn out to be useful for characterizing intermediate complexity classes, or considering an integrated semantical interpretation.

The importance and usefulness of the last point has recently been demonstrated by the work of Laurent and Tortora de Falco (Laurent and de Falco, 2005), who found a semantical characterization of elementary time by exploiting the syntactical results of Danos and Joinet.

The present work shows that Danos and Joinet's methodology, originally applied only to **ELL**, can be extended to **LLL**; in fact, we will see that it is possible to separate by purely structural means (extending the stratification condition) a subsystem of linear logic (with the addition of the § modality) which characterizes **FP**, the class of functions computable in polynomial time by a deterministic Turing machine. The definition of our subsystem is in the end very similar to that of **LAL**, even though the presence of additive connectives makes things a little more complicated; still, it arguably simplifies Girard's original definition of **LLL**. Moreover, we prove a polystep strong normalization result analogous to that proved by Terui for his light-affine λ-calculus (Terui, 2002): the evaluation of a program inside our system is always polynomially bounded, no matter what cut-elimination strategy is chosen (modulo some minor details concerning additive reductions).

The polytime system we define actually does not fully correspond to Girard's **LLL**: there are important differences at the level of additive connectives (the "exponential isomorphism" no longer holds). We address this issue by showing how the "original" **LLL** can be recovered inside linear logic (still with the addition of the paragraph modality) through an enhancement of Danos and Joinet's ideas. In passing, we also show how Lafont's Soft Linear Logic (Lafont, 2004) can be redefined in a similar way.

From the theoretical point of view, as already pointed out above, our work may offer the possibility of extending Laurent and Tortora de Falco's ideas to obtain a semantics for **LLL** based on some subset of their *obsessional cliques*.

From the point of view of applications, the absence of any intuitionistic restriction may turn out to be interesting: for example, Girard (Girard, 1998) remarked that it is possible to exploit the symmetries like $A \multimap B \simeq B^\perp \multimap A^\perp$ to program more "clever" algorithms than those accessible when considering only λ-terms (and thus intuitionism). This is of course impossible in **LAL**, since one cannot really work outside of its intuitionistic fragment (not doing so would immediately pose non-confluence problems).

Acknowledgments. Many, many special thanks go to Lorenzo Tortora de Falco, whose support has been fundamental to the development of this work. We would also like to thank the anonymous referee for his/her comments and suggestions.

2. Preliminaries

Linear logic proofs can be presentend using two alternative syntaxes: sequent calculus or proof-nets. In the present work, we shall largely prefer the latter, mainly because we are concerned with the complexity of the cut-elimination procedure within (subsystems of) linear logic. When the dynamics of cut-elimination is at study, the use of proof-nets becomes virtually mandatory, cut-elimination in sequent calculus being blurred by countless commuting convertions of no computational value.

More generally, we believe that proof-nets should be seen as the standard syntax for linear logic, and we see sequent calculus as a link to tradition, a connection to other logical systems in which proof-nets are not available. This is why we shall nevertheless formulate all of our results also in sequent calculus, with the double benefit that it will give us a convenient link to the λ-calculus (see Sect. 2.4) and perhaps bring our work to a larger audience.

2.1. *Syntax of* **LL** *and* $\mathbf{LL}_\S$

We start by recalling the basic syntactical definitions for second order linear logic without neutrals, which we refer to simply as **LL**. Let X range over a denumerably infinite set of propositional atoms, together with their negations $X^\perp$. The formulas of **LL**, ranged over by $A, B, \ldots$, are generated by the following grammar:

$$A, B ::= X \mid X^\perp \mid A \otimes B \mid A \parr B \mid A \oplus B \mid A \,\&\, B \mid \exists X A \mid \forall X A \mid !A \mid ?A$$

The negation $A^\perp$ of a formula A is defined as follows:

$$\begin{aligned} (X)^\perp &= X^\perp & (X^\perp)^\perp &= X \\ (A \otimes B)^\perp &= B^\perp \parr A^\perp & (A \parr B)^\perp &= B^\perp \otimes A^\perp \\ (A \oplus B)^\perp &= B^\perp \,\&\, A^\perp & (A \,\&\, B)^\perp &= B^\perp \oplus A^\perp \\ (\exists X A)^\perp &= \forall X A^\perp & (\forall X A)^\perp &= \exists X A^\perp \end{aligned}$$

$$(!A)^\perp = ?A^\perp \qquad (?A)^\perp = !A^\perp$$

We will also consider the additional modality § (*paragraph*), which is self-dual:

$$(\S A)^\perp = \S A^\perp$$

We call $\mathbf{LL}_\S$ the system obtained by adding the paragraph modality to **LL**.

2.2. *Proof-nets for* **LL** *and* $\mathbf{LL}_\S$

In this section we briefly recall the main definitions concerning proof-nets. The presentation we choose here is a variant of that proposed by Girard (Girard, 1996):

— like Girard, we decompose the introduction of a why not formula in two steps: first we make a *discharged formula* $\flat A$ ($[A]$ in Girard's notation; our notation is the one used by Lorenzo Tortora de Falco and Olivier Laurent for the polarized fragment of linear logic (Laurent and de Falco, 2004)), then we take several (maybe zero) occurrences of $\flat A$ and form $?A$. A discharged formula is not a formula; in particular, it cannot be composed by means of any connective.

— unlike Girard, we use *boxes* to treat additive conjunctions, as done by Tortora de Falco (Tortora de Falco, 2003). Even though absolutely against the spirit of proof-nets, this is arguably the simplest formulation in terms of correctness criteria, and since the additive connectives do not play a fundamental role in light logics, we believe it to be the most convenient one for our purposes.

In the following definition, and throughout the rest of the paper, unless explicitly stated we shall make confusion between the concepts of *formula* and *occurrence of formula*. The same will be done for links and their occurrences.

Definition 2.1 (Proof-structure). A *proof-structure* is a triple $(\mathcal{G}, \mathsf{B}, J)$, where $\mathcal{G}$ is a finite graph-like object whose nodes are occurrences of what we call *links*, and whose edges are directed and labelled by formulas or discharged formulas of **LL** (or $\mathbf{LL}_\S$); B is a set of subgraphs of $\mathcal{G}$ called *boxes*; J is a partial function from the links of $\mathcal{G}$ to sets of links of $\mathcal{G}$, called *jumps*.

— Each link has an arity and co-arity, which are resp. the number of its incoming and outgoing edges. The arity and co-arity is fixed for all links except *why not* links, which have co-arity 1 and arbitrary arity. A zeroary *why not* link is also referred to as a *weakening* link. *Par*, *for all*, and *why not* links are called *jumping* links; J is defined only on these links.

— The incoming edges of a link (and the formulas that label them) are referred to as its *premises*, and are supposed to be ordered, with the exception of *cut* and *why not* links; the outgoing edges of a link (and the formulas that label them) are referred to as its *conclusions*.

— Premises and conclusions of links must respect a precise labeling (which depends on the link itself), given in Fig. 1. In particular:

   – edges labelled by discharged formulas can only be premises of *pax*, *pad*, and *why not* links;

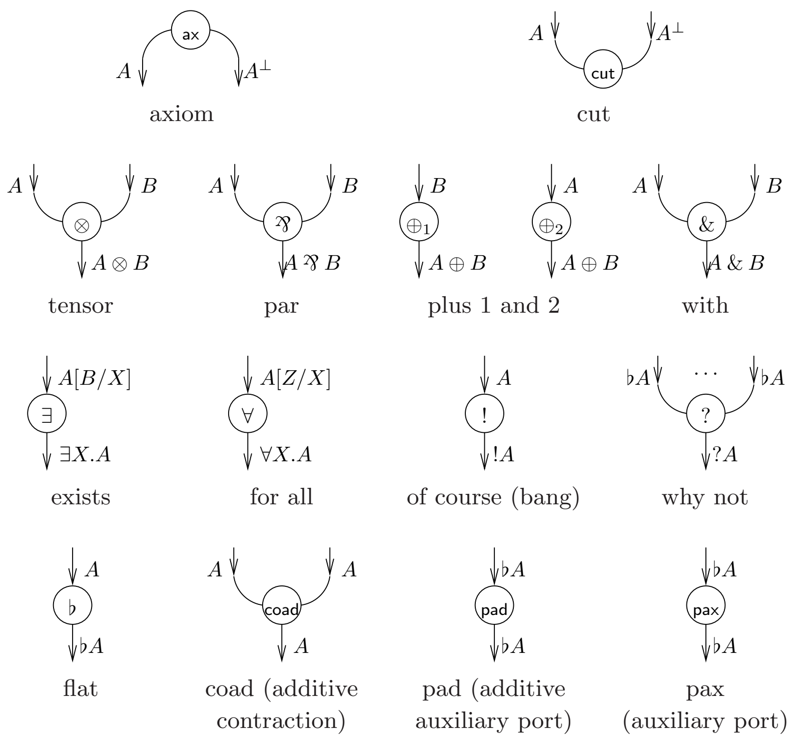


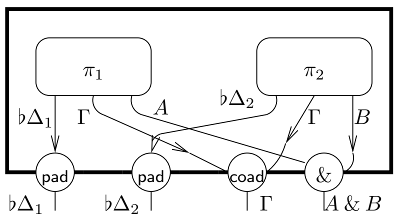
Fig. 1. Links

– in a *for all* link n, the variable Z in its premise $A[Z/X]$ is called the *eigenvariable* of n. Each *for all* link is supposed to have a different eigenvariable.

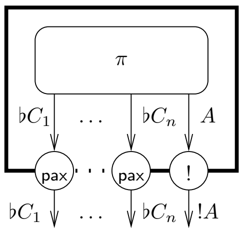
– in an *exists* link n, the formula B in its premise $A[B/X]$ is said to be *associated* to n.

— Each edge must be the conclusion of exactly one link, and the premise of at most one link, except the case of an edge labelled by a discharged formula or by a formula containing an eigenvariable, which must be the premise of exactly one link. Those edges that are not premises of any link (and the formulas that label them) are deemed *conclusions* of the proof-structure. The presence of these "pending" edges, together with the fact that some premises are ordered, is why proof-structures are not exactly graphs.

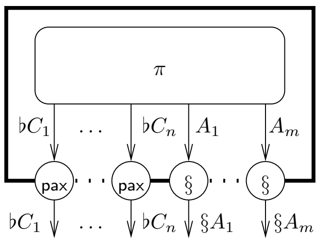
— Boxes can be of two types: *additive boxes* (Fig. 2a) and !-*boxes* (Fig. 2b). In the two figures, π_1, π_2, π are proof-structures, said to be *contained* in their respective box; in particular, π_1 and π_2 are said to be the two *components* of their additive box. The links that are explicitly represented (i.e., the with, coad and pad links in Fig. 2a and the bang and pax links in Fig. 2b) form the *border* of their respective box. The unique with (resp. of course) link in the border of an additive box (resp. !-box) is called the principal port of the box; the other links of the border, of which there may be an arbitrary number, are called additive contractions (coad) and additive auxiliary ports



a. Additive box



b. !-box



c. §-box

Fig. 2. Boxes

(pad) in the case of an additive box, or simply auxiliary ports (pax) in the case of a !-box. Boxes must have the following properties:

a. each with (resp. of course) link is the principal port of exactly one additive box (resp. !-box);

b. each coad or pad (resp. pax) link belongs to exactly one additive box (resp. !-box);

c. any two distinct boxes are either disjoint or included one in the other.

— Proof-structures for $\mathbf{LL}_§$ may contain another kind of link, the paragraph link, and a third kind of box, called §-box (Fig. 2c). §-boxes must have property *c* above, while in their presence property *b* becomes "each pax link belongs to exactly one !- *or* §-box". On the other hand, there is no notion of principal port for §-boxes. So while paragraph links must belong to a unique §-box, a §-box need not have any paragraph link, provided it has in this case at least one pax, i.e., the border can never be empty. In the context of $\mathbf{LL}_§$ proof-structures, we use the terminology *exponential box* to mean either !- or §-box.

— For each jumping link n, the set $J(n)$ (called the *jumps* of n) is defined as follows:

Par: $J(n)$ is the set containing the link(s) whose conclusions are the premises of n.

For all: if Z is the eigenvariable of n, $J(n)$ is the set containing:

- the link whose conclusion is the premise of n;
- any link whose conclusion is labelled by a formula containing Z;
- any *exists* link whose associated formula contains Z.

Why not: $J(n)$ is the set containing the link(s) whose conclusions are the premises of n, plus an axiom link a of $\mathcal{G}$, called *default jump* of n, such that if $\mathcal{B}$ is a box

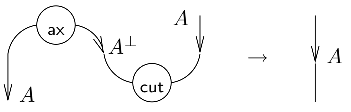


Fig. 3. Axiom step

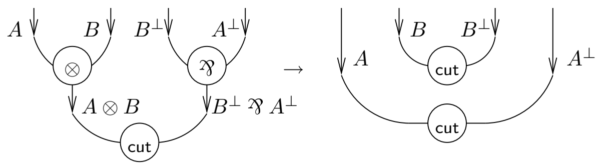


Fig. 4. Multiplicative step

(of any kind) of B and n is contained in $\mathcal{B}$, then a is contained in $\mathcal{B}$ (in the same component of n in case $\mathcal{B}$ is an additive box).

Definition 2.2 (Depth, sizes). Let σ be an **LL** or $\mathbf{LL}_\S$ proof-structure.

— A link of σ is said to have *depth d* if it is contained in d (necessarily nested) exponential boxes. The depth of a box of σ is the depth of the links forming its border. The depth of σ, denoted $\partial(\sigma)$, is the maximum depth of its links.
— The *partial size at depth d* of σ, denoted $|\sigma|_d$ is the number of links of σ different from *cut* links having depth d. The *size* of σ, denoted $|\sigma|$, is the sum of its partial sizes. The partial sizes and the size of a !-box $\mathcal{B}$ of σ, still denoted $|\mathcal{B}|_d$ and $|\mathcal{B}|$, are the partial sizes and the size of the proof-structure it contains.

Definition 2.3 (Switching). Let σ be an **LL** or $\mathbf{LL}_\S$ proof-structure. A *switching* of σ is an undirected graph built as follows:

— the conclusions of σ are erased, and its edges considered as undirected;
— for each jumping link n, the premises of n are erased, exactly one node $m \in J(n)$ is chosen and a new edge between m and n is added.
— the boxes at depth zero of σ are collapsed into single nodes, i.e., if $\mathcal{B}$ is a box at depth zero of σ, it is erased together with all the edges connecting its links to the rest of the structure, and replaced with a new node n; then, for any link m of depth zero which was connected to a link of $\mathcal{B}$, a new edge between m and n is added.

Definition 2.4 (Proof-net). A proof-net is a proof-structure $(\mathcal{G}, \mathsf{B}, J)$ such that:

— all of its switchings are acyclic and connected;
— if $\mathcal{B}$ is an additive box of B, and if $\mathcal{G}_1, \mathcal{G}_2$, $\mathsf{B}_1, \mathsf{B}_2$, and J_1, J_2 are the restrictions of $\mathcal{G}$, B, and J to the two respective components of $\mathcal{B}$, then $(\mathcal{G}_1, \mathsf{B}_1, J_1)$ and $(\mathcal{G}_2, \mathsf{B}_2, J_2)$ are proof-nets;
— if $\mathcal{B}$ is an exponential box of B, and if $\mathcal{G}_1$, B_1, and J_1 are the restrictions of $\mathcal{G}$, B, and J to the content of $\mathcal{B}$, then $(\mathcal{G}_1, \mathsf{B}_1, J_1)$ is a proof-net.

In the proof-net syntax, cut-elimination becomes a graph rewriting process. The cut-elimination steps are given in Figures 3 through 8. The only missing case is that of a cut

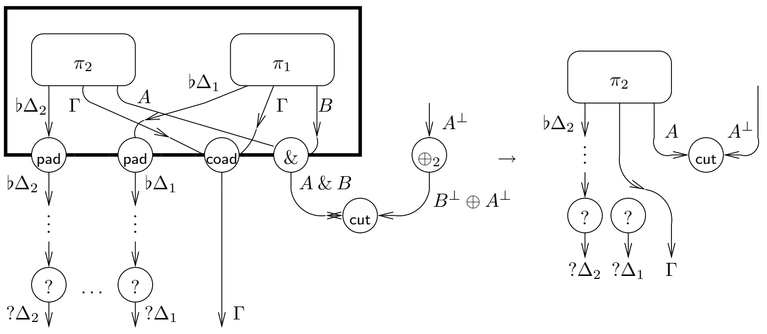


Fig. 5. Additive step (the case $\oplus_1$ is symmetrical)

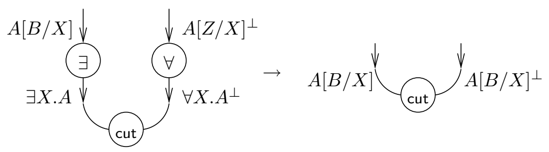


Fig. 6. Quantifier step

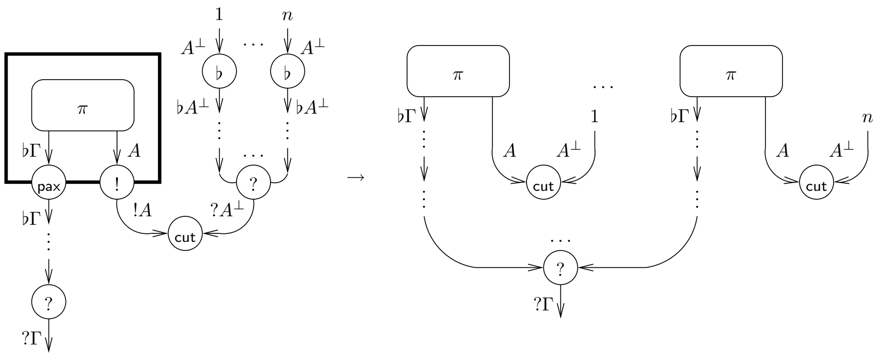


Fig. 7. Exponential step

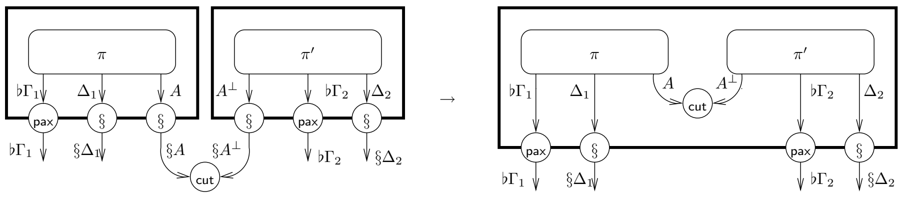


Fig. 8. Paragraph step

in which one of the premises is the conclusion of a coad link; such a cut is called *additive commutative cut*, and its reduction poses non-confluence problems. For this reason, we do not reduce these cuts, and we restrict to *lazy* normalization. When a proof-net π reduces to π' after the application of one cut-elimination step, we write $\pi \to \pi'$.

The graph-rewriting rules given in the figures above are not enough though. Whenever $\pi \to \pi'$, one must also take care of defining the jumps of π'; in fact, while the jumps for *par*, *for all*, and the non-default jumps for *why not* links of π' are forced by its structure, default jumps are *a priori* arbitrary, and must therefore be given by the cut-elimination procedure.

In order to properly describe what happens after each cut-elimination step, we need to define the notions of *lift* and *residue* of a link, as done by Tortora de Falco (Tortora de Falco, 2003):

Definition 2.5 (Lift, residue). Whenever $\pi \to \pi'$, by simple inspection of the cut-elimination rules it is clear that any link l' of π' different from a *cut* comes from a unique ("the same") logical link l of π; we say that l is the *lift* of l', and that l' is a *residue* of l. We define the lift and residues of a box (of any kind) in the same way.

Now let π be a proof-net such that $\pi \to \pi'$, let w' be a why not link of π', let w be its lift, whose default jump is j, and let R be the set of the residues of j in π'. The default jump j' of w' is defined depending on the nature of the step leading from π to π':

Axiom step: let a and c be resp. the axiom and cut link involved in the step. If $j \neq a$, then $R = \{j'_1\}$, and we set $j' = j'_1$. If $j = a$, then $R = \emptyset$, and we must find a new default jump. Now, there always exists a directed path (in plain graph-theoretical sense) starting from an axiom or why not link l of π and ending into c without passing through a (there may indeed be several). Let l' be the (only) residue of l in π'. If l is an axiom, we set $j' = l'$. If l is a why-not link, let j_1 be the default jump of l. Since π is a proof-net, $j_1 \neq a$ (otherwise there would be a cyclic switching), so j_1 has exactly one residue j'_1 in π'; we set $j' = j'_1$.

Multiplicative step: This step does not "touch" axioms, so $R = \{j'_1\}$, and we pose $j' = j'_1$.

Additive step: Let $\mathcal{B}$ be the additive box involved in the step. Notice that no duplications are made, so the cardinality of R is at most 1. If $R = \{j'_1\}$, we pose $j' = j'_1$. If $R = \emptyset$, it means that j belonged to the component of $\mathcal{B}$ erased by the step; in this case, let b be an axiom link belonging to the component *not* erased by the step (by correctness, there is at least one), and let b' be its residue in π'. We set $j' = b'$.

Quantifier step: As in the multiplicative step.

Exponential step: If $R \neq \emptyset$, we set j to be an arbitrary link of R. Otherwise, j has been erased, which means that we are dealing with a weakening step. So let us call $\mathcal{B}$ and w resp. the !-box and the weakening link involved. Let j_1 be the default jump of w; by acyclicity of all switchings, j_1 cannot be an axiom link contained in $\mathcal{B}$, therefore it has exactly one residue j'_1 in π'. We set $j' = j'_1$.

Paragraph step: As in the multiplicative step.

The following result assures us that the reassignment of default jumps defined above is correct:

Theorem 2.1 (Stability under cut-elimination (Girard, 1996)). Let π be a proof-net such that $\pi \to \pi'$. Then, π' is a proof-net.

This brief presentation of proof-nets will hopefully be enough to follow the rest of our work. For any further detail on proof-nets, we refer to the previously mentioned papers of Girard, Tortora de Falco, and Laurent (Girard, 1996; Tortora de Falco, 2003; Laurent and de Falco, 2004).

2.3. *Sequent calculus*

The sequent calculus of **LL** is defined by the following rules (as usual, Γ, Δ range over finite lists of formulas):

— **Identity rules**:

$$\frac{}{\vdash A^\perp, A}\,(\text{axiom}) \qquad \frac{\vdash \Gamma, A \qquad \vdash \Delta, A^\perp}{\vdash \Gamma, \Delta}\,(\text{cut})$$

— **Exchange** (the only structural rule):

$$\frac{\vdash \Gamma, A, B, \Delta}{\vdash \Gamma, B, A, \Delta}\,(\text{exchange})$$

— **Multiplicative rules**:

$$\frac{\vdash \Gamma, A \qquad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}\,(\text{tensor}) \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\,(\text{par})$$

— **Additive rules**:

$$\frac{\vdash \Gamma, A_i}{\vdash \Gamma, A_1 \oplus A_2}\,(\text{plus } i) \quad i \in \{1,2\} \qquad \frac{\vdash \Gamma, A \qquad \vdash \Gamma, B}{\vdash \Gamma, A \,\&\, B}\,(\text{with})$$

— **Quantifier rules**:

$$\frac{\vdash \Gamma, A[B/X]}{\vdash \Gamma, \exists X A}\,(\text{exists}) \qquad \frac{\vdash \Gamma, A}{\vdash \Gamma, \forall X A}\,(\text{for all}) \quad X \text{ not free in } \Gamma$$

— **Exponential rules**:

$$\frac{\vdash ?\Gamma, A}{\vdash ?\Gamma, !A}\,(\text{promotion}) \qquad \frac{\vdash \Gamma, A}{\vdash \Gamma, ?A}\,(\text{dereliction})$$

$$\frac{\vdash \Gamma}{\vdash \Gamma, ?A}\,(\text{weakening}) \qquad \frac{\vdash \Gamma, ?A, ?A}{\vdash \Gamma, ?A}\,(\text{contraction})$$

The sequent calculus of $\mathbf{LL}_\S$ is defined by adding the rule

$$\frac{\vdash ?\Gamma, \Delta}{\vdash ?\Gamma, \S\Delta}\,(\text{paragraph})$$

to the sequent calculus of **LL**. Notice that Δ may be empty; the necessity of this apparently useless "do-nothing" version of the rule is a consequence of the self-duality of

§, and becomes clear when considering cut-elimination under the stratification condition (Definition 3.4).

In some of our definitions we shall make use of the notion of *descendant* of an occurrence of formula in a sequent calculus derivation. Any **LL** (or $\mathbf{LL}_\S$) sequent calculus rule different from exchange, with, and contraction has the following shape:

$$\frac{\vdash \Gamma_1, A_1 \quad \dots \quad \vdash \Gamma_n, A_n}{\vdash \Gamma_1, \dots, \Gamma_n, A}$$

Therefore, any occurrence B_1 of a formula B in the conclusion of the rule, except the occurrence of A, "comes from" a unique occurrence B_0 of the same formula in one of the premises; we say that B_1 is an *immediate descendant* of B_0. Even though the exchange rule does not quite fit in the above scheme, any occurrence of formula in the conclusion of such a rule still "comes from" a unique occurrence of the same formula in the premise, so the definition of immediate descendant is identical. In a with rule, let B_1 be an occurrence of B in the context of the conclusion, and let B_0' (resp. B_0'') be the unique occurrence of B in the left (resp. right) premise from which B_1 "comes from". Then, B_1 is the immediate descendant of both B_0' and B_0''. Similarly, in a contraction rule deriving $\vdash \Gamma, ?A$ from $\vdash \Gamma, ?A, ?A$, all occurrences of formulas in the conclusion descend immediately from exactly one occurrence in the premise, except for the occurrence of $?A$, which is the immediate descendant of both occurrences of $?A$ in the premise. Now, if A' and A'' are two occurrences of the same formula A appearing in a sequent calculus derivation, we say that A'' is a descendant of A' (or that A' has A'' among its descendants) iff there exist n occurrences $A_1, \dots, A_n$ of A such that $A_1 = A'$, $A_n = A''$ and, for all i, A_{i+1} is the immediate descendant of A_i.

A proof-structure is *sequentializable* if it can be built inductively following the rules of sequent calculus. Proof-nets and sequent calculus derivations are linked by this fundamental result:

Theorem 2.2 (Girard, 1996). A proof-structure is sequentializable iff it is a proof-net.

Actually, since our proof-structures use discharged formulas, sequentialization is done in a sequent calculus in which all rules are as above except for additive conjunction, which becomes

$$\frac{\vdash \flat\Delta_1, \Gamma, A \qquad \vdash \flat\Delta_2, \Gamma, B}{\vdash \flat\Delta_1, \flat\Delta_2, \Gamma, A \mathbin{\&} B}\ (\text{with})$$

and the four exponential rules are replaced by

$$\frac{\vdash \flat\Gamma, A}{\vdash \flat\Gamma, !A}\ (\text{of course}) \qquad \frac{\vdash \Gamma, \flat A, \dots, \flat A}{\vdash \Gamma, ?A}\ (\text{why not}) \qquad \frac{\vdash \Gamma, A}{\vdash \Gamma, \flat A}\ (\text{flat})$$

with the proviso that discharged formulas are not principal formulas of any rule except the why not rule. But the reader may check that $\vdash \Gamma$ (with no discharged formulas in Γ) is provable in this modified sequent calculus iff $\vdash \Gamma$ is provable in the "standard" one, and since proof-structures are not allowed to have discharged formulas among their conclusions, Theorem 2.2 is already meaningful for the "standard" sequent calculus. This sequent calculus with discharged formulas will turn out to be useful in Sect. 6.

2.4. *Intuitionistic sequent calculus and term assignment*

It is also possible to consider an asymmetric version of the calculus above, which yields what is usually called *intuitionistic* linear logic. Negation is never considered in this system, so there are only positive variables $X, Y, \ldots$, and formulas are given by

$$A, B ::= X \mid A \otimes B \mid A \multimap B \mid A \,\&\, B \mid \forall X A \mid !A$$

The only interest of the intuitionistic calculus is that it can be decorated by λ-terms, as done by Danos and Joinet (Danos and Joinet, 2003). Here, we need to add some constructs to the λ-calculus in order to handle the tensor connective. If x ranges over a denumerably infinite set of variables, we start by extending variables into *patterns*:

$$\mathbf{x}, \mathbf{y} ::= x \mid \mathbf{x} \otimes \mathbf{y}$$

The terms of our calculus are then defined as follows:

$$t, u ::= x \mid \lambda \mathbf{x}.t \mid tu \mid t \otimes u \mid \langle t, u \rangle \mid \mathsf{fst}\ t \mid \mathsf{snd}\ t$$

We consider application to be associative to the left, i.e., tuv is short for $(tu)v$, whereas both patterns and tensors of terms are associative to the right, i.e., $x \otimes (y \otimes z)$ and $t \otimes (u \otimes v)$ are simply written resp. as $x \otimes y \otimes z$ and $t \otimes u \otimes v$. Moreover, terms of the form

$$\lambda \mathbf{x} \otimes z.(\lambda \mathbf{y}.t)z$$

are written more concisely as

$$\lambda \mathbf{x} \otimes \mathbf{y}.t\ ,$$

which can be seen as a sort of η-reduction.

A term is said to *match* a pattern if it is a tensor of terms with the same associative structure as the pattern. More formally, any term matches the pattern x, while t matches $\mathbf{x} \otimes \mathbf{y}$ iff $t = u \otimes v$ and u matches $\mathbf{x}$ and v matches $\mathbf{y}$. For example, $t \otimes u \otimes v$ matches $x \otimes y \otimes z$, but not $(x \otimes y) \otimes z$, unless $t = t_1 \otimes t_2$. Substitution can then be extended to patterns and matching terms:

— $t[u/x]$ is defined as usual;
— $t[u \otimes v/\mathbf{x} \otimes \mathbf{y}] = t[u/\mathbf{x}, v/\mathbf{y}]$.

Terms are equipped with the following reduction rules:

$$\text{if } u \text{ matches } \mathbf{x}, \text{ then} \quad (\lambda \mathbf{x}.t)u \to t[u/\mathbf{x}]$$

$$\mathsf{fst}\ \langle t, u \rangle \to t$$

$$\mathsf{snd}\ \langle t, u \rangle \to u$$

In case no rule can be applied, a term is said to be *normal*. For example, if t is normal, then so is $(\lambda x \otimes y.t)z$.

The term assignment for the intuitionistic version of **LL** is defined as follows:

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

$$\frac{\Gamma \vdash t : A \qquad \Delta, x : B \vdash u : C}{\Gamma, \Delta, y : A \multimap B \vdash u[yt/x] : B}\ y \text{ “fresh”} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \multimap B}$$

$$\frac{\Gamma \vdash t : A \qquad \Delta \vdash u : B}{\Gamma, \Delta \vdash t \otimes u : A \otimes B} \qquad \frac{\Gamma, x : A, y : B \vdash t : C}{\Gamma, z : A \otimes B \vdash (\lambda x \otimes y.t)z : C}\ z \text{ “fresh”}$$

$$\frac{\Gamma, x : A \vdash t : C}{\Gamma, y : A \,\&\, B \vdash t[\mathsf{fst}\ y/x]}\ y \text{ “fresh”} \qquad \frac{\Gamma, x : B \vdash t : C}{\Gamma, y : A \,\&\, B \vdash t[\mathsf{snd}\ y/x]}\ y \text{ “fresh”}$$

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash u : B}{\Gamma \vdash \langle t, u\rangle : A \,\&\, B}$$

$$\frac{\Gamma, x : A[B/X] \vdash t : C}{\Gamma, x : \forall X A \vdash t : C} \qquad \frac{\Gamma \vdash t : A}{\Gamma \vdash t : \forall X A}\ X \text{ not free in } \Gamma$$

$$\frac{!\Gamma \vdash t : A}{!\Gamma \vdash t : !A} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma, x : !A \vdash t : B}$$

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

The assignment is compatible with normalization:

Proposition 2.3. If $\mathcal{D}$ is an intuitionistic derivation reducing to $\mathcal{D}'$ through cut-elimination, and if t, t' are the λ-terms assigned resp. to $\mathcal{D}$ and $\mathcal{D}'$, then $t \to^* t'$.

The intuitionistic sequent calculus has therefore a very concrete computational semantics, which turns out to be quite useful when one needs to program inside **LL**. Indeed, we will rely upon it to prove the completeness of our polytime system. Notice also that everything we said can be extended without problems to $\mathbf{LL}_\S$: we simply need to add the typing judgment

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

Of course, it is always possible for an intuitionistic derivation to be translated in the one-sided sequent calculus; it is enough to replace $A \multimap B$ with $A^\perp ⅋ B$ and to negate the formulas on the left side of the sequent. In particular, one can always see an intuitionistic derivation of $\Gamma \vdash A$ as a proof-net of conclusions $\Gamma^\perp, A$.

3. The system $\overline{\mathbf{LLL}}$

3.1. *Definition*

We start by briefly recalling the definition of **ELL**, which is based upon the so-called *stratification condition.* The presentation chosen by Danos and Joinet (Danos and Joinet, 2003) formulates the condition using **LL** sequent calculus:

Definition 3.1 (ELL, sequent calculus (Danos and Joinet, 2003)). *Elementary Linear Logic* is the subsystem of **LL** composed of all the derivations satisfying the following *stratification condition*: any occurrence of a why not formula $?A$ introduced by a dereliction rule (resp. an axiom rule) has exactly one (resp. zero) descendant(s) in the context of a promotion rule.

The definition can be adapted to proof-nets:

Definition 3.2 (Structural branch). A *structural branch* of an **LL** or $\mathbf{LL}_\S$ proof-net is a directed path (in the graph-theoretical sense) starting from a *flat* link and ending into a *why not* link.

Definition 3.3 (ELL, proof-nets). **ELL** is the subsystem of **LL** composed of all the proof-nets π such that any structural branch of π crosses exactly one *pax* link.

The main result concerning **ELL** is the following:

Theorem 3.1 (Danos and Joinet, 2003). **ELL** is sound and complete with respect to the class of Kalmar elementary functions.

In order to transport Danos and Joinet's approach to polynomial time, we shall follow Girard's recommendation, which is "to restrict promotion to a unary context":

Definition 3.4 ($\overline{\mathbf{LLL}}$, proof-nets). $\overline{\mathbf{LLL}}$ is the subsystem of $\mathbf{LL}_\S$ composed of all the proof-nets π which satisfy the following conditions:

i. Each !-box of π has at most one auxiliary port.
ii. Any structural branch of π crosses exactly one *pax* link.

Of course, condition **ii** is simply Danos&Joinet's stratification, extended to §-boxes as well.

For the sake of completeness, we also give an equivalent definition in terms of sequent calculus:

Definition 3.5 ($\overline{\mathbf{LLL}}$, sequent calculus). $\overline{\mathbf{LLL}}$ is the subsystem of $\mathbf{LL}_\S$ composed of all the derivations satisfying the following conditions:

i. The context of each promotion rule contains at most one formula.
ii. Any occurrence of a why not formula $?A$ introduced by a dereliction rule (resp. an axiom rule) has exactly one (resp. zero) descendant(s) in the context of a promotion or a paragraph rule.
iii. Any occurrence of a why not formula $?A$ appearing in the context of a promotion rule is the descendant of at most one occurrence of $?A$ introduced by a dereliction rule.

Obviously, conditions **i** and **ii** are straight-forward rephrasings of the two conditions of Definition 3.4; in particular, **ii** is Danos&Joinet's stratification condition extended to paragraph rules, and in this context we refer to it as the *downward* stratification condition.

Condition **iii**, called *upward* stratification condition, is needed because of the redundant sequential information contained in a sequent calculus derivation (and is yet another witness of how proof-nets are such a better syntax to work with!). Basically, it is needed to reject derivations like

$$\cfrac{\cfrac{\cfrac{\vdash A, A, B}{\vdash ?A, ?A, B}}{\vdash ?A, B}}{\vdash ?A, !B}$$

which respects conditions **i** and **ii** (assuming the sub-derivation ending with $\vdash A, A, B$ does), but is clearly "cheating".

3.2. *Cut-elimination*

Cut-elimination inside $\overline{\mathbf{LLL}}$ is defined as in $\mathbf{LL}_\S$ (Figures 3 through 8). It takes then a simple case-by-case inspection to verify the following:

Proposition 3.2. Let π be an $\overline{\mathbf{LLL}}$ proof-net of depth d, and let $\pi \to \pi'$. Then:

1 π' is in $\overline{\mathbf{LLL}}$ as well;

2 $\partial(\pi') \leq d$, and the inequality may be strict only if the cut-elimination step leading from π to π' is an additive step or an exponential step involving a weakening link.

The presence of second order quantification requires some care though. As a matter of fact, it is not hard to show that the iterated substitution of a formula to a propositional variable can cause phenomena of exponential growth in the size of the formulas typing the proof-net, which is quite harmful if one wants to implement *polytime* cut-elimination on a Turing machine. But, as already noticed by Girard, we can solve the problem by forgetting types during reduction; once reached the normal form, the typing can be correctly recovered if the conclusions of the proof-net contain no existential quantifier.

This means that we work with untyped lazy normalization, so we are guaranteed to reach a well-typed cut-free proof-net only if the conclusions of the starting proof-net are &- and $\exists$-free:

Definition 3.6 (Lazy proof-net). An $\mathbf{LL}_\S$ proof-net is said to be *lazy* if it does not contain additive conjunctions or existential quantifiers in its conclusions.

Theorem 3.3 (Tortora de Falco, 2003). Cut-elimination for lazy proof-nets is confluent and strongly normalizing.

The restriction to lazy proof-nets will not be a problem with respect to the expressive power of $\overline{\mathbf{LLL}}$, since in the completeness proof we shall use only types which respect this constraint.

4. Cut-elimination bounds

The aim of this section is to improve Theorem 3.3 by showing that not only all reduction sequences starting with an $\overline{\mathbf{LLL}}$ proof-net terminate, but they do so in a number of steps which is polynomial in the size of the starting proof-net. We refer to this property as

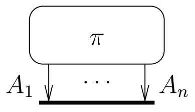


Fig. 9. A garbage island; π is a generic proof-net.

polystep strong normalization; it was first proved by Kazushige Terui for his light-affine λ-calculus, which is based on intuitionistic light affine logic (Terui, 2002).

Our proof roughly applies the same argument used by Terui, although the presence of additive connectives requires some technical adjustments. The proof unfolds as follows:

(a) We first define a technical variant of the "standard" cut-elimination procedure, in order to have a subtler handling of *erasing*, especially that introduced by additive steps. This alternative cut-elimination procedure (called *GC-cut-elimination*, "GC" standing for *garbage collection*) is shown to be no better than the standard one in terms of the length of the longest reduction sequence normalizing an $\overline{\mathbf{LLL}}$ proof-net (Proposition 4.2, Sect. 4.1).

(b) Using GC-cut-elimination, we can normalize a proof-net by first applying only non-erasing reduction steps at non-decreasing depth, and perform the garbage collection, i.e., the erasing, only when there is nothing else left to do. Reduction sequences following this "protocol" are called *slow*, and they are shown to be indeed the worst in terms of length (Slowdown Lemma, Sect. 4.2). This is intuitively justified by the fact that a slow reduction sequence never tries to do work before duplicating it and never refrains from performing useless work.

(c) Using an argument similar to that of Girard (Girard, 1998), we prove that for any $\overline{\mathbf{LLL}}$ proof-net π there exists an explicit bound to the length of its slow reduction sequences, and thus, by point (b), of all possible reduction sequences starting from π. This bound is polynomial in the size of π, and doubly-exponential in its depth (Theorem 4.9, Sect. 4.4). Although formulated in terms of GC-cut-elimination, by point (a) the bound is valid for "standard" cut-elimination as well, and the proof of polystep strong normalization is then complete.

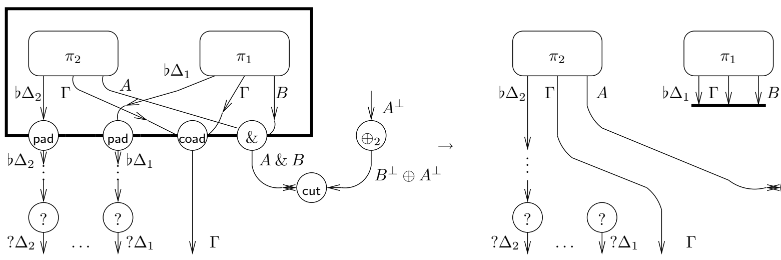
4.1. *Cut-elimination with explicit garbage collection*

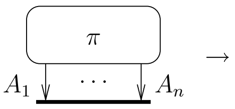
For technical reasons, it turns out to be necessary to slightly modify the "standard" cut-elimination procedure described and used in Sections 2 and 3. Basically, we shall make explicit the *garbage collection* process in cut-elimination.

Definition 4.1 (Garbage island, GC-proof-net). A *garbage island* is an $\mathbf{LL}_\S$ proof-net the conclusions of which have been "killed" by a special link, called *garbage link*, of arbitrary arity and coarity zero (Fig. 9). A *GC-proof-net* is an $\mathbf{LL}_\S$ proof-net as in Definition 2.4 plus any number of (necessarily unconnected) garbage islands.

Of course it makes sense to speak of $\overline{\mathbf{LLL}}$ GC-proof-nets, as the constraints of Definition 3.4 can be applied just as they are to $\mathbf{LL}_\S$ GC-proof-nets. The same applies to *lazy* GC-proof-nets. The definitions of depth, size, etc. also remain unchanged.

The cut-elimination procedure is extended to GC-prof-nets by modifying the additive cut-elimination step, which will be responsible for the introduction of garbage islands, and by adding a further step, called *garbage collection step*, which simply removes a garbage island:





(the additive step involving a $\oplus_1$ link is defined symmetrically). All other cut-elimination steps remain defined as in Sect. 2.2. The cut-elimination procedure for GC-proof-nets will be called *GC-cut-elimination.* Notice that a garbage island may contain cuts, which can be reduced *before* applying a garbage collection step. The addition of this useless work is actually the reason behind the introduction of GC-proof-nets.

GC-cut-elimination is confluent and strongly normalizing (cut-free in the case of lazy GC-proof-nets), and Proposition 3.2 holds for GC-cut-elimination as well, with the fundamental difference that, in point 2, the depth of a proof-net can no longer decrease after an additive step but after a garbage collection step. Moreover, we clearly have the following:

Proposition 4.1.

1 The normal form of a GC-proof-net under GC-cut-elimination is always a proof-net, i.e., it contains no garbage island.

2 Let π be an $\mathbf{LL}_\S$ proof-net. Then, the normal form of π found through the "standard" cut-elimination procedure and the normal form found through GC-cut-elimination coincide.

If π is an $\mathbf{LL}_\S$ proof-net, $\|\pi\|$ will denote the length of the longest GC-cut-elimination sequence starting from π. When we want to speak at the same time of the two cut-elimination procedures, the notation $\|\pi\|_{\mathsf{std}}$ will be used to refer to the "standard" one.

Definition 4.2 (Standard reduction sequence). A sequence of GC-cut-elimination steps is called *standard* if every additive step is followed by a garbage collection step.

Standard reduction sequences are nothing but sequences in the… "standard" cut-elimination procedure, i.e., the set of standard reduction sequences between π and π' is in bijection with the reduction sequences between π and π' in the "standard" cut-elimination procedure; the only difference is that each additive step counts as two steps. We thus have the following:

Proposition 4.2. If π is an $\mathbf{LL}_\S$ proof-net, then $\|\pi\|_{\mathsf{std}} \leq \|\pi\|$.

Proof. Simply observe that, by the above remark, given (one of) the longest reduction sequence(s) starting from π in the "standard" cut-elimination procedure, there is a standard GC-cut-elimination sequence starting from π of at least the same length. $\square$

Propositions 4.1 and 4.2 basically assure us that we can use GC-cut-elimination as a tool for a worst-case analysis of the runtime of an $\overline{\mathbf{LLL}}$ proof-net under the "standard" cut-elimination procedure.

For this reason, in the following sections we shall speak of "proof-nets" and "cut-elimination" actually meaning "GC-proof-nets" and "GC-cut-elimination", while it should be clear that whenever we state a runtime bound, we implicitly (and most importantly) state it also for the "standard" cut-elimination procedure.

4.2. *Slow reduction sequences*

Let us start by introducing some useful notations and terminology. In the sequel, *instances* of cut-elimination steps and the steps themselves will be systematically confused.

— $\mathbf{LL}_\S$ cut-elimination steps (or, more simply, steps) will be ranged over by e. Reduction sequences are words over steps, i.e., if $e_1, \ldots, e_n$ are steps, $S = e_1 \cdots e_n$ is a reduction sequence, the length of which is denoted by $|S|$;

— if an $\mathbf{LL}_\S$ proof-net π reduces to π' through a reduction sequence S, we write $\pi \overset{S}{\longrightarrow^*} \pi'$;

— we use the notation $e \in S$ to say that e is a step contained in the reduction sequence S. Two steps $e, e' \in S$ are naturally ordered by $e \leq_S e'$ iff e is applied before or is equal to e' in S. We use the notation $e <^1_S e'$ to indicate that e is applied *immediately before* e';

— suppose that $S = S_1 \cdot e' \cdot e \cdot S_2$, more precisely

$$\pi \overset{S_1}{\longrightarrow^*} \pi_1 \overset{e'}{\longrightarrow} \pi_2 \overset{e}{\longrightarrow} \pi_3 \overset{S_2}{\longrightarrow^*} \pi'.$$

We say that e *pre-commutes* with e' if we can find another reduction sequence $S' = S_1 \cdot e \cdot S'' \cdot S_2$ such that

$$\pi \overset{S_1}{\longrightarrow^*} \pi_1 \overset{e}{\longrightarrow} \pi_2' \overset{S''}{\longrightarrow^*} \pi_3 \overset{S_2}{\longrightarrow^*} \pi'$$

where S'' is a non-empty sequence containing only copies of e'. In such a case, we obviously have $|S| \leq |S'|$;

— the depth of a step e, denoted $\partial(e)$, is the depth of the cut link reduced by e;

— we call *non-erasing* a reduction sequence in which neither garbage collection steps nor exponential steps involving a weakening link are performed;

Definition 4.3 (Monotonic reduction sequence). A reduction sequence S is said to be *monotonic* iff for any two cut-elimination steps $e, e' \in S$, $e \leq_S e'$ implies $\partial(e) \leq \partial(e')$.

Definition 4.4 (Slow reduction sequence). A reduction sequence S is *slow* iff $S = M \cdot E$, where M is a monotonic non-erasing reduction sequence and E contains only garbage collection steps or exponential steps involving weakening links.

Lemma 4.3 (Slowdown). For any $\mathbf{LL}_\S$ proof-net π there exists a slow reduction sequence S (lazy-)normalizing π such that $|S| = \|\pi\|$.

Proof. Let π' be the cut-free form of π, and let S' be a non-slow reduction sequence leading to it. We shall "slow down" S' by transforming it into a slow reduction sequence S such that $\pi \overset{S}{\rightarrow^*} \pi'$ and $|S'| \leq |S|$, which is enough to prove our statement.

First of all, observe that if $e' <^1_{S'} e$ and e' is a garbage collection or weakening step, then e always pre-commutes with e' without altering the reduction length. In other words, erasing steps can always be "delayed" and pushed toward the end of the reduction sequence.

It is then enough to consider a non-erasing S', and show that if $e, e' \in S'$ are such that $e' <^1_{S'} e$ but $\partial(e) < \partial(e')$, then e pre-commutes with e', and the monotonicity is locally established. By iterating this procedure, we obviously obtain a globally monotonic sequence of length at least $|S'|$.

We start by observing that any cut-elimination step can only affect the depths greater or equal to its own, and by hypothesis $\partial(e) < \partial(e')$; this means that the cut reduced by e is "already there" when e' is executed. We can then always execute e first, and since by hypothesis it is non-erasing, after its execution we are left with at least one copy of the cut involved in e', which means that e always pre-commutes with e'.

For what concerns the monotonicity, observe that the execution of e can affect the cut c reduced by e' only if e is an exponential step; but even in this case, the depths of the copies of c produced by e are no smaller that $\partial(e)$, which means that the sequence is locally monotonic. $\square$

The Slowdown Lemma is very useful: it tells us that in a worst-case analysis, it is enough to consider slow reduction sequences.

4.3. *Legs and contractive proof-nets*

Let us introduce yet another class of reduction sequences:

Definition 4.5 (k-leg). A non-erasing reduction sequence S is a *k-leg* iff for all $e \in S$, $\partial(e) = k$. A k-leg S is *stretched* iff whenever $e \in S$ is an exponential step and $e' \in S$ is a non-exponential step, $e' \leq_S e$. A k-leg is *complete* iff it ends with a proof-net which is lazy-cut-free at depth k.

Lemma 4.4 (Stretching). Let S be a k-leg such that $\pi \overset{S}{\rightarrow^*} \pi'$. Then, there exists a stretched k-leg S' such that $\pi \overset{S'}{\rightarrow^*} \pi'$ and $|S'| = |S|$.

Proof. Any step at depth k pre-commutes with an exponential step at the same depth, without altering the length of the reduction. $\square$

Definition 4.6 (k-contractive proof-net). A proof-net π is said to be *k-contractive* iff all cuts at depth k in π are exponential.

Notice that executing a step at depth k in a k-contractive proof-net yields another k-contractive proof-net, i.e., k-legs preserve "k-contractiveness".

From here on, we only consider $\overline{\mathbf{LLL}}$ proof-nets. By the stratification condition, any premise of a why not link l of an $\overline{\mathbf{LLL}}$ proof-net must belong to a structural branch crossing an auxiliary port of exactly one exponential box $\mathcal{C}$; we then say that $\mathcal{C}$ is *above* l. Additionally, the notation $\pi^{(k)}$ is used to denote the portion of a proof-net π of depth k, i.e., when we refer to a link or a box of $\pi^{(k)}$, we are talking about a link or a box having depth k in π.

Definition 4.7 (Contractive order). Let π be a k-contractive $\overline{\mathbf{LLL}}$ proof-net. If $\mathcal{B}$ and $\mathcal{C}$ are two !-boxes of $\pi^{(k)}$, we write $\mathcal{B} \lhd^1_\pi \mathcal{C}$ iff there is a cut of $\pi^{(k)}$ whose premises are the conclusion of the principal port of $\mathcal{B}$ and the conclusion of a why not link l such that $\mathcal{C}$ is above l. The reflexive-transitive closure of $\lhd^1_\pi$, denoted $\unlhd_\pi$, is called the *contractive order* of $\pi^{(k)}$.

In the above definition, $\unlhd_{\pi^{(k)}}$ would be a heavier but more rigorous notation; we omit the depth because we never consider two contractive orders of two different depths in the same proof-net.

Proposition 4.5. $\unlhd_\pi$ is an arborescent partial order.

Proof. The anti-symmetry of $\unlhd_\pi$ is a consequence of the acyclicity of proof-nets. For what concerns the arborescence, simply consider that any $\overline{\mathbf{LLL}}$!-box $\mathcal{B}$ has at most one auxiliary port, which means that $\mathcal{C} \lhd^1_\pi \mathcal{B}$ for at most one $\mathcal{C}$. □

Intuitively, $\mathcal{B} \unlhd_\pi \mathcal{C}$ means that at some point in the reduction of $\pi^{(k)}$ a copy of the content of $\mathcal{B}$ will be "poured" into $\mathcal{C}$. The maximal elements of the contractive order are !-boxes which are either not involved in any non-erasing step, or whose content is entirely "poured" into §-boxes. Notice that Proposition 4.5 is false in **ELL**, i.e., the order is not arborescent; this is the ultimate reason behind the polystep bounds holding for $\overline{\mathbf{LLL}}$.

If $\mathcal{B}$ is a !-box involved in an exponential cut with a why-not link l, in the following we denote by $\nabla(\mathcal{B})$ the arity of l minus the number of !-boxes above l. If the conclusion of the principal port of $\mathcal{B}$ is not the premise of a cut link, we shall conventionally set $\nabla(\mathcal{B}) = 1$.

Definition 4.8 (Contractive factor, potential size). If π is a k-contractive $\overline{\mathbf{LLL}}$ proof-net, the *contractive factor* of an exponential box $\mathcal{B}$ of $\pi^{(k)}$, denoted $\mu(\mathcal{B})$, is defined as follows:

- ▶ if $\mathcal{B}$ is a §-box, then $\mu(\mathcal{B}) = 1$.
- ▶ if $\mathcal{B}$ is a !-box, then $\mu(\mathcal{B}) = \sum_{\mathcal{B} \unlhd_\pi \mathcal{C}} \nabla(\mathcal{C})$.

For all i such that $k < i \leq \partial(\pi)$, the *potential size at depth i* of an exponential box $\mathcal{B}$ of $\pi^{(k)}$, denoted $[\mathcal{B}]_i$, is the product of its partial size at depth i and its contractive factor:

$$[\mathcal{B}]_i = \mu(\mathcal{B}) \cdot |\mathcal{B}|_i \ .$$

The *potential size at depth i of* π, also denoted $[\pi]_i$, is the sum of all the potential sizes at depth i of all exponential boxes of $\pi^{(k)}$.

The contractive factor has the following properties:

Lemma 4.6. Let π be a k-contractive $\overline{\mathbf{LLL}}$ proof-net, and $\mathcal{B}$ a !-box of $\pi^{(k)}$. Then:

1 $\mu(\mathcal{B}) \leq |\pi|_k$;
2 $\mu(\mathcal{B}) = \nabla(\mathcal{B}) + \sum_{\mathcal{B} \lhd^1_\pi \mathcal{C}} \mu(\mathcal{C})$.

Proof. For what concerns point 1, simply observe that, for any !-box $\mathcal{C}$ of $\pi^{(k)}$, $\nabla(\mathcal{C})$ is at most equal to the arity of some why not link at depth k, unique for each $\mathcal{C}$. Now, a why not link of arity n at depth k in $\overline{\mathbf{LLL}}$ requires the presence of n pax links at the same depth, therefore the sum of the arities of all why not links cannot exceed $|\pi|_k$.

Point 2 is a consequence of Proposition 4.5:

$$\mu(\mathcal{B}) = \sum_{\mathcal{B} \unlhd_\pi \mathcal{C}} \nabla(\mathcal{C}) = \nabla(\mathcal{B}) + \sum_{\mathcal{B} \lhd^1_\pi \mathcal{C}} \sum_{\mathcal{C} \unlhd_\pi \mathcal{D}} \nabla(\mathcal{D}) = \nabla(\mathcal{B}) + \sum_{\mathcal{B} \lhd^1_\pi \mathcal{C}} \mu(\mathcal{C}) \ .$$

□

Lemma 4.7. Let π be a k-contractive $\overline{\mathbf{LLL}}$ proof-net, and let i be such that $k < i \leq \partial(\pi)$. Then:

1 if π is lazy-cut-free at depth k, then $[\pi]_i = |\pi|_i$;
2 if π reduces to π' after executing one non-erasing step at depth k, then $[\pi']_i = [\pi]_i$.

Proof. First observe that if π does not contain cuts at depth k, its contractive order is trivial, i.e., for each !-box $\mathcal{B}$, $\mathcal{B}$ is maximal and moreover $\nabla(\mathcal{B}) = 1$. Therefore, $\mu(\mathcal{B}) = 1$ for any exponential box of $\pi^{(k)}$, so $[\mathcal{B}]_i = |\mathcal{B}|_i$ and $[\pi]_i = |\pi|_i$.

For what concerns point 2, let $\mathcal{B}$ and l be resp. the !-box and the why not link (by hypothesis of arity $n \geq 1$) involved in the cut-elimination step, and let $\mathcal{C}_1, \ldots, \mathcal{C}_n$ be the exponential boxes above l. If $\nabla(\mathcal{B}) = m$, we can assume that $\mathcal{C}_1, \ldots, \mathcal{C}_m$ are §-boxes, and $\mathcal{C}_{m+1}, \ldots, \mathcal{C}_n$ are !-boxes, so that, using Lemma 4.6,

$$\mu(\mathcal{B}) = m + \sum_{j=m+1}^{n} \mu(\mathcal{C}_j).$$

The execution of the step does not concern any box other than $\mathcal{B}$ and $\mathcal{C}_1, \ldots, \mathcal{C}_n$. More precisely:

— $\mathcal{B}$ has no residue; its content is copied and "poured" into $\mathcal{C}_1, \ldots, \mathcal{C}_n$.
— $\mathcal{C}_1, \ldots, \mathcal{C}_n$ have exactly one residue, which we call resp. $\mathcal{C}'_1, \ldots, \mathcal{C}'_n$.
— For all i such that $k < i \leq \partial(\pi)$, $|\mathcal{C}'_j|_i = |\mathcal{B}|_i + |\mathcal{C}_j|_i$.
— $\nabla(\mathcal{C}'_j) = \nabla(\mathcal{C}_j)$.
— Apart from the removal of $\mathcal{B}$, the contractive order is unchanged: if $\mathcal{D}'$ is the residue of $\mathcal{D}$, $\mathcal{C}'_j \lhd^1_{\pi'} \mathcal{D}'$ iff $\mathcal{C}_j \lhd^1_\pi \mathcal{D}$.
— As a consequence of the last two points above, $\mu(\mathcal{C}'_j) = \mu(\mathcal{C}_j)$.

Now, the potential size at depth i of π' is the sum of the $[\mathcal{C}'_j]_i$ plus the sum of the potential sizes of those boxes not touched by the step, which we call c:

$$[\pi']_i \, = c + \sum_{j=1}^{n} [\mathcal{C}'_j]_i = c + \sum_{j=1}^{n} \mu(\mathcal{C}'_j)|\mathcal{C}'_j|_i = c + \sum_{j=1}^{n} \mu(\mathcal{C}_j)(|\mathcal{B}|_i + |\mathcal{C}_j|_i) =$$

$$= c + \sum_{j=1}^{n} \mu(\mathcal{C}_j)|\mathcal{B}|_i + \sum_{j=1}^{n}[\mathcal{C}_j]_i = c + (m + \sum_{j=m+1}^{n} \mu(\mathcal{C}_j))|\mathcal{B}|_i + \sum_{j=1}^{n}[\mathcal{C}_j]_i =$$

$$= c + \mu(\mathcal{B})|\mathcal{B}|_i + \sum_{j=1}^{n}[\mathcal{C}_j]_i = c + [\mathcal{B}]_i + \sum_{j=1}^{n}[\mathcal{C}_j]_i = [\pi]_i$$

which proves point 2 of our statement. □

4.4. *Runtime bound*

Lemma 4.8. Let π be an $\overline{\mathbf{LLL}}$ proof-net of depth d such that $\pi \overset{S}{\to^*} \pi'$, where S is a complete stretched k-leg. Then, if $s_0, \ldots, s_d$ are the partial sizes of π, the partial sizes of π' are at most $s_0, \ldots, s_k, s_k s_{k+1}, \ldots, s_k s_d$. Moreover, $|S| \leq s_k$.

Proof. First of all notice that since S is non-erasing, by point 2 of Proposition 3.2 $\partial(\pi') = d$, which means that there are indeed exactly d partial sizes for π', let us call them s'_i, $0 \leq i \leq d$.

Now, since nothing is done at depth $i \neq k$, and since whenever the content of a box is duplicated, the only partial sizes affected are those at depth greater than that of the box, we have that for all $i < k$, $s'_i = s_i$.

The fact that S is stretched means that $S = N \cdot X$, where N contains only non-exponential steps and X only exponential steps. Non-exponential steps do not alter the partial sizes at depth different than k (not even paragraph steps), and strictly shrink the size at depth k. Therefore, after executing N, we have a k-contractive proof-net of partial sizes at most equal to those of π.

During the execution of X, the size continues to shrink at depth k, so s'_k cannot be bigger than s_k; this also proves that $|S| \leq s_k$. On the other hand, at depth $i > k$, we can use Lemmas 4.6 and 4.7 to obtain

$$s'_i = [\pi]_i = \sum_{\mathcal{B} \in \pi^{(k)}} [\mathcal{B}]_i = \sum_{\mathcal{B} \in \pi^{(k)}} \mu(\mathcal{B})|\mathcal{B}|_i \leq s_k \sum_{\mathcal{B} \in \pi^{(k)}} |\mathcal{B}|_i = s_k s_i \ .$$

□

Theorem 4.9 (Polystep strong normalization). Let π be an $\overline{\mathbf{LLL}}$ proof-net of size s and depth d. Then, $\|\pi\| \leq 2s^{2^d}$.

Proof. By the Slowdown Lemma 4.3, we know that there is a slow reduction sequence S normalizing π in $\|\pi\|$ steps. By definition, we have $S = M \cdot E$, where M is monotone non-erasing, and E is made of weakening and garbage collection steps only. But by monotonicity, and by point 2 of Proposition 3.2, $M = L_0 \cdots L_d$, where each L_k is a (maybe empty) complete k-leg. Moreover, by the Stretching Lemma 4.4, we can assume L_k to be stretched.

Now, if $s_0, \ldots, s_d$ are the partial sizes of π, by Lemma 4.8 we know that $|L_0| \leq s_0$, yielding a proof-net of partial sizes at most $s_0, s_0 s_1, \ldots, s_0 s_d$; in the same way, $|L_1| \leq s_0 s_1$, yielding a proof-net of partial sizes $s_0, s_0 s_1, s_0^2 s_1 s_2, \ldots, s_0^2 s_1 s_d$, and so on. In general, we have

$$|L_k| \leq s_k \prod_{i=0}^{k-1} s_{k-i-1}^{2^i} \ , \quad 0 \leq k \leq d \ .$$

Therefore, using the fact that the partial sizes cannot be greater than the total size, the length of M can be bounded as follows:

$$\begin{aligned} |M| &= \sum_{k=0}^{d} |L_k| \leq \sum_{k=0}^{d} s_k \prod_{i=0}^{k-1} s_{k-i-1}^{2^i} \leq \sum_{k=0}^{d} s_k \prod_{i=0}^{k-1} s^{2^i} = \\ &= \sum_{k=0}^{d} s_k s^{\sum_{i=0}^{k-1} 2^i} = \sum_{k=0}^{d} s_k s^{2^k-1} \leq s^{2^d-1} \sum_{k=0}^{d} s_k = s^{2^d} \ . \end{aligned}$$

An identical calculation shows that the proof-net obtained at the end of M has total size at most s^{2^d}, which means that $|E| \leq s^{2^d}$ (the size strictly decreases under the execution of an erasing step). So $\|\pi\| = |S| = |M| + |E| \leq 2s^{2^d}$. □

5. Polytime soundness and completeness

We proceed now with the proofs of soundness and completeness of $\overline{\mathbf{LLL}}$ with respect to the class **FP**. Actually, the hard part of proving soundness has already been done in the previous section; here we just need to define exactly how we represent functions inside $\overline{\mathbf{LLL}}$. Some care is needed since the bound proved is polynomial *in the size*, but can be as bad as doubly-exponential in the depth! Fortunately, $\overline{\mathbf{LLL}}$ is not very different from the other "light" systems, and the standard types and encodings used for **LLL** and **LAL** work just fine.

In particular, the main technical issue concerning completeness, which is encoding the transition of a Turing machine in such a highly constrained language as $\overline{\mathbf{LLL}}$, is addressed by suitably reworking Luca Roversi's argument for **LAL** (Roversi, 1999). Therefore, our proof does not contain any noteworthy novelty at all, and we shall be brief on it.

It must also be mentioned that, following a recent work of Harry Mairson and Kazushige Terui (Mairson and Terui, 2003), the additive-free fragment of Girard's **LLL** is **FP**-time complete. Now, since **LLL** without additives is clearly a subsystem of $\overline{\mathbf{LLL}}$, no completeness proof would actually be needed, which is a further reason why we can avoid showing it in full detail. In fact, our encoding (which by the way uses the additives) is sketched here solely for the sake of self-containedness.

5.1. *Representing functions on integers and strings*

In order to get closer to usual functional languages and types, we shall shift to the intuitionistic framework (Sect. 2.4), and replace multiplicative disjunction with linear implication, defined as $A \multimap B := A^{\perp} \parr B$.

Using the abbreviation $A^n = A \otimes \cdots \otimes A$, we define

$$\mathbf{S}_k = \forall X.(!(X \multimap X))^k \multimap \S(X \multimap X)$$

as the type of finite strings over an alphabet of k symbols. The cases $\mathbf{S}_1$ and $\mathbf{S}_2$ correspond resp. to unary integers and binary strings, and are abbreviated resp. $\mathbf{N}$ and $\mathbf{S}$. The terms attached to the derivations of $\vdash \mathbf{N}$ are the Church integers, denoted $\underline{n}$; an example of a term attached to a derivation of $\vdash \mathbf{S}$ is $\lambda s_1 \otimes s_0.\lambda z.s_1(s_1(s_0(s_1 z)))$, representing the binary string 1011 and therefore denoted $\underline{1011}$.

We now formalize the notion of *representation* inside $\overline{\mathbf{LLL}}$ for the crucial cases of functions from binary strings to binary strings and from natural numbers to natural numbers. In the following, $\S^p A$ is short for $\S \ldots \S A$, where A is preceded by p paragraph modalities.

Definition 5.1 (Representation). A function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ (resp. $f : \mathbb{N}^k \rightarrow \mathbb{N}$) is *representable* inside $\overline{\mathbf{LLL}}$ if there exist a non-negative integer p and an intuitionistic derivation $\underline{f}$ of the sequent $\mathbf{S} \vdash \S^p\mathbf{S}$ (resp. $\mathbf{N}^k \vdash \S^p\mathbf{N}$) such that, for any binary string x (resp. k non-negative integers $n_1, \ldots, n_k$), $f(x) = y$ (resp. $f(n_1, \ldots, n_k) = m$) iff the proof obtained by cutting the proof $\underline{x}$ of the sequent $\vdash \mathbf{S}$ (resp. the tensor of the proofs $\underline{n_1}, \ldots, \underline{n_k}$ of $\vdash \mathbf{N}$) to $\underline{f}$ normalizes to a proof of $\vdash \S^p\mathbf{S}$ (resp. $\vdash \S^p\mathbf{N}$) which is $\underline{y}$ (resp. $\underline{m}$) with the addition of p paragraph rules at the end. We shall denote by $\mathbf{F}\overline{\mathbf{LLL}}$ the set of functions from binary strings to binary strings representable inside $\overline{\mathbf{LLL}}$.

5.2. FP-*soundness*

A fundamental remark is that the proof-nets of conclusion $\mathbf{S}$ have all depth 1 and size proportional to the length of the binary string they represent. Therefore, if φ is a proof-net of conclusions $\mathbf{S}^\perp, \S^p\mathbf{S}$, say of size s and depth d, the proof-net π obtained by cutting any $\underline{x}$ of conclusion $\mathbf{S}$ with φ has depth $c = \max\{d, 1\}$ and size $k|x| + m$, where k and m are suitable constants (m depending on s). By Theorem 4.9, there exists a polynomial P of degree depending on c (and so depending solely on φ) such that the normal form of π is reached in less than $P(k|x| + m) = O(P(|x|))$ steps. Since each reduction step at most squares the size of a proof-net, implementing reduction on a Turing machine can be done at worst with a polynomial slowdown, so we get

Theorem 5.1 (Polytime soundness). Every function from binary strings to binary strings representable inside $\overline{\mathbf{LLL}}$ is computable in polynomial time by a deterministic Turing machine, i.e. $\mathbf{F}\overline{\mathbf{LLL}} \subseteq \mathbf{FP}$.

5.3. FP-*completeness*

The rest of the section is devoted to sketching the proof of the converse of Theorem 5.1:

Theorem 5.2 (Polytime completeness). Every function from binary strings to binary strings computable in polynomial time by a deterministic Turing machine is representable inside $\overline{\mathbf{LLL}}$, i.e. $\mathbf{FP} \subseteq \mathbf{F}\overline{\mathbf{LLL}}$.

In the following, we will use expressions of the form

$$\mathsf{name}[x_1, \ldots, x_n] : C_1, \ldots, C_n \vdash A$$

to say that name is a proof of the sequent $C_1, \dots, C_n \vdash A$ whose decoration is a λ-term (also called name) of type A and containing the free variables $x_1, \dots, x_n$ of respective types $C_1, \dots, C_n$.

Iteration, concatenation/addition, and polynomials. Given k proofs $\mathsf{step}_i : C_i, A \vdash A$, $1 \leq i \leq k$ (where C_i does not need to be there), and a proof $\mathsf{base} : \Delta \vdash A$, we can define the iteration scheme $\mathsf{it}^k_{\mathsf{step}_1,\dots,\mathsf{step}_k,\mathsf{base}}[x]$ as follows (we write $\mathbf{S}_k[A]$ for the instantiation of the second order quantifier in $\mathbf{S}_k$ over the formula A):

$$\begin{prooftree}
\AxiomC{$\vdots\ \mathsf{step}_1$}
\noLine
\UnaryInfC{$C_1, A \vdash A$}
\UnaryInfC{$C_1 \vdash A \multimap A$}
\UnaryInfC{$!C_1 \vdash A \multimap A$}
\UnaryInfC{$!C_1 \vdash !(A \multimap A)$}
\AxiomC{$\dots$}
\AxiomC{$\vdots\ \mathsf{step}_k$}
\noLine
\UnaryInfC{$C_k, A \vdash A$}
\UnaryInfC{$C_k \vdash A \multimap A$}
\UnaryInfC{$!C_k \vdash A \multimap A$}
\UnaryInfC{$!C_k \vdash !(A \multimap A)$}
\doubleLine
\TrinaryInfC{$!C_1, \dots, !C_k \vdash (!(A \multimap A))^k$}
\AxiomC{$\vdots\ \mathsf{base}$}
\noLine
\UnaryInfC{$\Delta \vdash A$}
\AxiomC{}
\UnaryInfC{$A \vdash A$}
\BinaryInfC{$\Delta, A \multimap A \vdash A$}
\doubleLine
\UnaryInfC{$!\Delta', \Delta'', A \multimap A \vdash A$}
\UnaryInfC{$!\Delta', \S\Delta'', \S(A \multimap A) \vdash \S A$}
\BinaryInfC{$!C_1, \dots, !C_k, !\Delta', \S\Delta'', \mathbf{S}_k[A] \vdash \S A$}
\UnaryInfC{$!C_1, \dots, !C_k, !\Delta', \S\Delta'', \mathbf{S}_k \vdash \S A$}
\end{prooftree}$$

The attached term is $x(\mathsf{step}_1 \otimes \dots \otimes \mathsf{step}_k)\mathsf{base}$, where x is the free variable of type $\mathbf{S}_k$. Using the iteration scheme, we can program coercions as in (Girard, 1998), i.e. proofs of $\mathbf{S}_k \vdash \S^{p+1}!^q\mathbf{S}_k$ which represent the identity.

Now, if $A^{(n)}$ stands for n occurrences of the formula A, it is obvious that, for any $n \geq 1$ and any formula A, using n axioms and $n-1$ tensor rules, we can build a proof of $A^{(n)} \vdash A^n$. If we put $I := X \multimap X$, then we can consider the following derivation:

$$\begin{prooftree}
\AxiomC{$\vdots$}
\noLine
\UnaryInfC{$(!I)^{(p)} \vdash (!I)^p$}
\AxiomC{$\vdots$}
\noLine
\UnaryInfC{$(!I)^{(p)} \vdash (!I)^p$}
\AxiomC{}
\UnaryInfC{$X \vdash X$}
\AxiomC{}
\UnaryInfC{$X \vdash X$}
\AxiomC{}
\UnaryInfC{$X \vdash X$}
\BinaryInfC{$X, I \vdash X$}
\BinaryInfC{$X, I, I \vdash X$}
\UnaryInfC{$I, I \vdash I$}
\UnaryInfC{$\S I, \S I \vdash \S I$}
\doubleLine
\TrinaryInfC{$(!I)^{(2p)}, \mathbf{S}_p[X], \mathbf{S}_p[X] \vdash \S I$}
\doubleLine
\UnaryInfC{$(!I)^{(p)}, \mathbf{S}_p[X], \mathbf{S}_p[X] \vdash \S I$}
\doubleLine
\UnaryInfC{$\mathbf{S}_p[X], \mathbf{S}_p[X] \vdash \mathbf{S}_p[X]$}
\doubleLine
\UnaryInfC{$\mathbf{S}_p, \mathbf{S}_p \vdash \mathbf{S}_p[X]$}
\UnaryInfC{$\mathbf{S}_p, \mathbf{S}_p \vdash \mathbf{S}_p$}
\end{prooftree}$$

We shall call this proof $\mathsf{concat}_p[x, y]$; it is easy to see that $\mathsf{concat}_p[\underline{s}, \underline{t}]$ reduces to $\underline{st}$, i.e. it concatenates two strings.

The reader can check that in the case $p = 1$, the term attached to the proof is $\lambda s.\lambda z.ms(nsz)$, which is the standard term for addition on Church numerals; so we put $\mathsf{add}[m, n] := \mathsf{concat}_1[m, n]$. Using this, we can build the term $\mathsf{it}^1_{\mathsf{add},\underline{0}}[m, n] :!\mathbf{N}, \mathbf{N} \vdash \S\mathbf{N}$, from which by a paragraph rule and a cut with two suitable coercions we obtain $\mathsf{mul}[m,n] : \mathbf{N}, \mathbf{N} \vdash §^2\mathbf{N}$, which represents multiplication. Notice that mul cannot be iterated further, i.e. it cannot be used as the step in the iteration scheme, which is natural, since the contrary would allow us to program exponential functions.

It is not hard to see that out of add and mul we can build a proof $\underline{P}[n] : \mathbf{N} \vdash §^{d+5}\mathbf{N}$ representing any polynomial P of degree d with non-negative integer coefficients. Actually, substraction and division can also be programmed in $\overline{\mathbf{LLL}}$, and arbitrary polynomials can be built, but what we have done so far is already enough to represent polytime bounds.

Turing machines. Using the abbreviation $A^{\&n} = A \& \cdots \& A$, we first define the type of finite alphabets of k symbols as

$$\mathbf{F}_k = \forall X.X^{\&k} \multimap X \ .$$

Following the decoration given in Sect. 2.4, one sees for example that the only two terms of type $\mathbf{F}_2$ are $\lambda x.\mathsf{fst}x$ and $\lambda x.\mathsf{snd}x$, i.e., the two boolean values.

We then introduce the type of configurations of Turing machines with p symbols and q states:

$$\mathbf{T}_{p,q} = \forall X.(!(X \multimap X))^p \multimap §(X^2 \multimap (X^2 \otimes \mathbf{F}_q)) \ .$$

As an example, the following is the term representing the configuration of a Turing machine with 3 symbols (0, 1 and *blank*) and q states, in which the current state is represented by the λ-term $Q : \mathbf{F}_q$, the tape contains the string 10110, and the head is positioned on the leftmost 0:

$$\lambda s_1 \otimes s_0 \otimes s_b.\lambda l \otimes r.s_0(s_1 l) \otimes s_1(s_1(s_0 r)) \otimes Q \ .$$

Given a Turing machine M with 3 symbols (represented resp. by the proofs $\underline{0}$, $\underline{1}$ and $\underline{b}$) and q states, calculating some function f from binary strings to binary strings, we need a proof $\mathsf{trans}_M[t] : \mathbf{T}_{3,q} \vdash \mathbf{T}_{3,q}$ such that, whenever $\underline{T} : \mathbf{T}_{3,q}$ is a configuration of M, $\mathsf{trans}_M[\underline{T}]$ reduces to $\underline{T'}$ iff according to the transition table of M, the configuration T yields the configuration T'. This is the technically difficult part of the completeness proof; as already said, it is possible to address it using an adaptation of Roversi's solution (Roversi, 1999).

A crucial step is encoding a function that, given a string on the alphabet $\{0, 1, b\}$, returns at the same time the first character and the rest of the string. Such a function would correspond to a proof of $\mathbf{S}_3 \vdash \mathbf{F}_3 \otimes \mathbf{S}_3$; actually, for the purpose of encoding the transition of a Turing machine, a proof of $\mathbf{S}_3 \vdash \mathbf{F}_3 \otimes X$ sufficies, where X is a propositional variable.

We start by introducing an auxiliary type

$$\mathbf{H} = \mathbf{F}_3 \otimes (X \& X).$$

By means of a with and a tensor rule, we first build a derivation $\mathsf{base}[z]$ whose attached term is $\underline{b} \otimes \langle z, z \rangle$. Then, let $j \in \{0, 1, b\}$, let s_j be variables of type $I = X \multimap X$, and consider the trivial derivations $\langle s_j \rangle : I \vdash I^{\&3}$ and $\langle \mathsf{id} \rangle : \vdash I^{\&3}$, whose attached terms are resp. $\langle\langle s_j, s_j\rangle, s_j\rangle$ and $\langle\langle \lambda z.z, \lambda z.z\rangle, \lambda z.z\rangle$. Observe that, for any term c of type $\mathbf{F}_3$, $c\langle s_j\rangle \rightarrow^* s_j$ and $c\langle \mathsf{id}\rangle \rightarrow^* \lambda z.z$. Using these derivations, we build $\mathsf{fetch}_j[s_j]$ as follows (below, $\mathbf{F}_3[I]$ denotes the instantiation of the second order quantifer in $\mathbf{F}_3$ over the formula I, and both of the with rules applied to the left of sequents introduce the "left" occurrence of X in $X \,\&\, X$):

$$
\begin{prooftree}
\AxiomC{$\vdots\ \underline{j}$}
\noLine
\UnaryInfC{$\vdash \mathbf{F}_3$}
\AxiomC{$\vdots\ \langle s_j\rangle$}
\noLine
\UnaryInfC{$I \vdash I^{\&3}$}
\AxiomC{}
\UnaryInfC{$X \vdash X$}
\AxiomC{}
\UnaryInfC{$X \vdash X$}
\BinaryInfC{$I, X \vdash X$}
\UnaryInfC{$I, X \,\&\, X \vdash X$}
\BinaryInfC{$I, \mathbf{F}_3[I], X \,\&\, X \vdash X$}
\doubleLine
\UnaryInfC{$I, \mathbf{H} \vdash X$}
\UnaryInfC{$!I, \mathbf{H} \vdash X$}
\AxiomC{$\vdots\ \langle \mathsf{id}\rangle$}
\noLine
\UnaryInfC{$\vdash I^{\&3}$}
\AxiomC{}
\UnaryInfC{$X \vdash X$}
\AxiomC{}
\UnaryInfC{$X \vdash X$}
\BinaryInfC{$I, X \vdash X$}
\UnaryInfC{$I, X \,\&\, X \vdash X$}
\BinaryInfC{$\mathbf{F}_3[I], X \,\&\, X \vdash X$}
\doubleLine
\UnaryInfC{$\mathbf{H} \vdash X$}
\UnaryInfC{$!I, \mathbf{H} \vdash X$}
\BinaryInfC{$!I, \mathbf{H} \vdash X \,\&\, X$}
\BinaryInfC{$!I, \mathbf{H} \vdash \mathbf{H}$}
\UnaryInfC{$!I \vdash \mathbf{H} \multimap \mathbf{H}$}
\UnaryInfC{$!I \vdash !(\mathbf{H} \multimap \mathbf{H})$}
\end{prooftree}
$$

The derivation is in $\overline{\mathbf{LLL}}$, since the promotion rule applied at the end has only one formula in its context, and this formula descends from only one dereliction. The term attached to $\mathsf{fetch}_j[s_j]$ is

$$\lambda x.(\underline{j} \otimes \langle\ (\lambda c \otimes r.c\langle s_j\rangle \mathsf{fst}\ r)x\ ,\ (\lambda c \otimes r.c\langle \mathsf{id}\rangle \mathsf{fst}\ r)x\ \rangle)\ .$$

Together, base and fetch_j can be used to extract the head and tail of a list. As a matter of fact, we invite the reader to check that the term

$$\lambda s.(\lambda c \otimes p.c \otimes \mathsf{snd}\ p)(s(\mathsf{fetch}_1 \otimes \mathsf{fetch}_0 \otimes \mathsf{fetch}_b)\mathsf{base}[z]),$$

applied to a string on the alphabet $\{0, 1, b\}$ of the form $\lambda s_1 \otimes s_0 \otimes s_b.\lambda z.s_j S$ yields $\underline{j} \otimes S$, or $\underline{b} \otimes z$ if the list is empty. The latter behavior accounts for the case in which the head of the machine tries to read "beyond" the boundaries of the tape: a blank character is read, and the tape is "extended".

Notice that the additives play a key rôle in the implementation of the function; they are used in a way which is reminiscent of Danos and Joinet's encoding of the predecessor (Danos and Joinet, 2003).

The idea above is the key to the encoding of Turing machines: without going over the details, the current configuration is transformed into an object of type $\mathbf{F}_3 \otimes \mathbf{H} \otimes \mathbf{H}$, which contains the current state and resp. the "left" and "right" portions of the tape, with their first symbols (thus including the current one) already extracted. Then, the current state and symbol are fed into a lookup-table encoding the transition function (still implemented using the additives), and the new configuration is produced.

To conclude, we still need a few "helper functions" (again, we shall not show the derivations explicitly): a proof $\mathsf{len}[x] : \mathbf{S} \vdash \mathbf{N}$, which returns the length of a binary string, a proof $\mathsf{init}[x] : \mathbf{S} \vdash \mathbf{T}_{3,q}$ which, given a binary string x, yields the starting configuration of a Turing machine with the string x on the tape, and a proof $\mathsf{read}[t] : \mathbf{T}_{3,q} \vdash \S\mathbf{S}$ which, given a configuration, reads the string which is written on the tape. The attached terms are

$$\begin{aligned}
\mathsf{len}[x] &= \lambda s.\lambda z.xssz \\
\mathsf{init}[x] &= \lambda s_1 \otimes s_0 \otimes s_b.\lambda l \otimes r.l \otimes (x(s_1 \otimes s_0)r) \otimes Q_0 \\
\mathsf{read}[t] &= (\lambda r \otimes l \otimes k.\lambda s_1 \otimes s_0.\lambda z.k\langle \ldots \langle \mathsf{id}, \mathsf{id}\rangle \ldots \mathsf{id}\rangle \\
&\quad (r(s_1 \otimes s_0)(l(s_1 \otimes s_0)z))) \\
&\quad (t((\lambda x.\mathsf{concat}_2[x, \underline{1}])\otimes \\
&\quad (\lambda x.\mathsf{concat}_2[x, \underline{0}])\otimes \\
&\quad (\lambda x.\mathsf{concat}_2[x, \underline{0}]))(\underline{\varepsilon} \otimes \underline{\varepsilon}))
\end{aligned}$$

where $Q_0 : \mathbf{F}_q$ is the term representing the initial state of the Turing machine, $\mathsf{id} = \lambda x.x$ and $\underline{\varepsilon}$ is the term representing the empty binary string, i.e. $\lambda s_1 \otimes s_0.\lambda z.z$.

Let now M be a Turing machine with 3 symbols and q states whose runtime is bounded by a polynomial P (of degree d) in the length of the input string. With the iteration scheme, we build $\mathsf{it}^1_{\mathsf{trans}_M,\mathsf{init}}[x, m] :!\mathbf{S}, \mathbf{N} \vdash \S\mathbf{T}_{3,q}$, which takes a binary string x, an integer m, and yields the configuration resulting from the execution of m elementary transitions of M with initial input x.

Since M does not need more than $P(|x|)$ steps to get to the result, we can imagine m to be exactly the result of the evaluation of P applied to $|x|$; so we apply a few paragraph rules to get a proof of $\S^{d+5}!\mathbf{S}, \S^{d+5}\mathbf{N} \vdash \S^{d+6}\mathbf{T}_{3,q}$, which can be cut with $\underline{P}$ and a coercion to obtain $\mathbf{S}, \mathbf{N} \vdash \S^{d+6}\mathbf{T}_{3,q}$.

This latter proof can now be cut with len on the conclusion of type $\mathbf{N}$ to get $\mathsf{comp}'_M[x, y] : \mathbf{S}, \mathbf{S} \vdash \S^{d+6}\mathbf{T}_{3,q}$. It is evident that $\mathsf{comp}'_M[x, x]$ takes a binary string x and yields the configuration resulting from $P(|x|)$ transitions of M with initial input x, which is right what we want. So we just need to add a couple of derelictions and a paragraph rule, and then contract to get $\mathsf{comp}_M[x] :!\mathbf{S} \vdash \S^{d+7}\mathbf{T}_{3,q}$.

Our read can now easily be transformed into a proof of $\S^{d+7}\mathbf{T}_{3,q} \vdash \S^{d+8}\mathbf{S}$, which cut with comp_M yields a proof of $!\mathbf{S} \vdash \S^{d+8}\mathbf{S}$. One more paragraph rule and a cut with a coercion and we get $\underline{f} : \mathbf{S} \vdash \S^{d+9}\mathbf{S}$, which represents the function f computed by M.

Feeding a binary string x into f means cutting the proof representing x with $\underline{f}$; we thus obtain a proof π of conclusion $\vdash \S^{d+9}\mathbf{S}$. It is fundamental here to remark that the formula $\mathbf{S}$ does not contain additive conjunctions or existential quantifiers, which means that the proof-net corresponding to π is lazy (Definition 3.6). Therefore, Theorem 3.3 applies, and after eliminating the cuts we get the cut-free proof-net representing $f(x)$ “boxed” inside a certain number of §-boxes, as required by Theorem 5.2 and the definition of representability.

Additive connectives are thus used only during the computation, without appearing in the result; their convenience has already been pointed out in the encoding of the function described above, which splits a list into head and tail.

6. Other subsystems of LL/LL$_\S$

We have seen how the addition of a simple structural condition to Danos&Joinet's **ELL** (plus the addition of the paragraph modality) yields a logical system faithfully characterizing deterministic polytime computation, in which moreover the polynomial bound does not depend upon the reduction strategy. How does this system compare to existing "light logics"?

It is not hard to see that the multiplicative/exponential fragment of $\overline{\mathbf{LLL}}$ coincides with the corresponding fragment of **LLL**. The additive connectives behave quite differently though. The main diverging point is the lack of the "exponential isomorphism" in $\overline{\mathbf{LLL}}$: because of condition **i** of Definition 3.4, it is impossible to prove $!A \otimes !B \multimap !(A \mathbin{\&} B)$.

There are also minor differences in the other direction, i.e., formulas that $\overline{\mathbf{LLL}}$ validates but that fail in **LLL**, but they are all ascribable to the unprovability of $!1$ in **LLL**, a feature which Girard himself states to be "a matter of taste". Therefore, $\overline{\mathbf{LLL}}$ can be seen as a subsystem of **LLL**; except for the absence of free weakening, it looks more akin to Asperti's **LAL** than to Girard's original system. Essentially, it would be fair to say that **LAL** is the affine variant of $\overline{\mathbf{LLL}}$, and not of **LLL**.

6.1. *Light Linear Logic*

Is it possible to "strengthen" the additives of $\overline{\mathbf{LLL}}$ to obtain a system which truly corresponds to **LLL**? As we shall see, the answer is positive, but requires structural conditions which are more complicated to formulate than those of Definition 3.4. The reward is that we find a framework in which also other "light logics" (i.e., **ELL** and **SLL**) can be recovered.

Definition 6.1 (Weighed proof-net). Consider the commutative monoid

$$\mathcal{Q} = \left\{ \frac{m}{2^n} \mid m, n \in \mathbb{N} \right\} \cup \{\omega\},$$

where the extra element ω is such that, for all $q \in \mathcal{Q}$, $q + \omega = \omega$. If we define $\omega/2 = \omega$, $\mathcal{Q}$ is closed under division by 2. It is also a totally ordered set, as soon as we set $q \leq \omega$ for all $q \in \mathcal{Q}$.

Let now π be an **LL**$_\S$ proof-net, and let $\mathcal{S}$ be the set of the edges of π labelled by discharged formulas. We define a function $w : \mathcal{S} \to \mathcal{Q}$ as follows:

— if s is the conclusion of a *flat* link, $w(s) = 1$;
— if s, s' are resp. the premise and the conclusion of a *pad* link, $w(s') = \frac{w(s)}{2}$;
— if s is the conclusion of a *pax* link, $w(s) = \omega$.

Let $\mathcal{B}$ be an exponential box of π, and let $s_1, \ldots, s_n$ be the premises of its auxiliary ports. We define the weight of $\mathcal{B}$ as

$$w(\mathcal{B}) = \sum_{i=1}^{n} w(s_i).$$

Let $\mathcal{B}$ be an additive box of π, and let $s_1^1, \ldots, s_{n_1}^1$ and $s_1^2, \ldots, s_{n_2}^2$ be the premises of the *pad* links of $\mathcal{B}$ belonging resp. to the "left" and "right" component of $\mathcal{B}$, such that

$s_i^j \neq \omega$. We define the *left weight* of $\mathcal{B}$ as

$$\overleftarrow{w}(\mathcal{B}) = \sum_{i=1}^{n_1} w(s_i^1),$$

and the *right weight* of $\mathcal{B}$ as

$$\overrightarrow{w}(\mathcal{B}) = \sum_{i=1}^{n_2} w(s_i^2).$$

We remark that the (boolean) weights found in Girard's proof-nets (Girard, 1996) are completely unrelated to the ones defined above.

Danos&Joinet's stratification can be reformulated in terms of weights:

Definition 6.2 (ELL, weighed proof-nets). **ELL** is the subsystem of **LL** composed of all the proof-nets π such that:

(**4**) if s is the premise of a *why not* link of π, then $w(s) = \omega$;
(**T**) if $\mathcal{B}$ is a !-box of π, then $w(\mathcal{B}) < \omega$.

Condition (**4**) forbids dereliction; if we restrict to proof-nets satisfying it, we obtain the subsystem of **LL** that Danos and Joinet call **4LL**. Similarly, condition (**T**) forbids digging, and in its presence we obtain the subsystem called **TLL**. These two systems intersect in **ELL**, and this is why both conditions are required in the definition.

$\overline{\textbf{LLL}}$ can be defined by adding the restriction to at most one auxiliary port, but this constraint looks a little "out of style" in this context. In fact, the following is a much more natural definition:

Definition 6.3 (LLL, weighed proof-nets). **LLL** is the subsystem of $\mathbf{LL}_\S$ composed of all the proof-nets π such that:

(**4**) if s is the premise of a *why not* link of π, then $w(s) = \omega$;
(**L**) if $\mathcal{B}$ is a !-box (resp. §-box) of π, then $w(\mathcal{B}) = 1$ (resp. $w(\mathcal{B}) < \omega$);
(**A**) if $\mathcal{B}$ is an additive box of π, then $\overleftarrow{w}(\mathcal{B}) = \overrightarrow{w}(\mathcal{B})$.

Condition (**L**) implies condition (**T**), so the above system (if we do not consider the paragraph modality) is obviously a subsystem of **ELL**. It could be replaced by $w(\mathcal{B}) \leq 1$, which would allow the provability of !1 (if we had admitted neutrals), but we prefer to stick to Girard's original definition (Girard, 1998).

The fact that Definition 6.3 actually defines a subsystem, i.e., a set of proof-nets stable under cut-elimination, is not trivial; as a matter of fact, condition (**A**) plays a crucial role in this respect.

Lemma 6.1. Let π be an **LLL** proof-net, $\mathcal{B}$ an additive box of π, and let $s_1, \ldots, s_n$ be the conclusions of the *pad* links of $\mathcal{B}$ such that $s_i \neq \omega$. Then,

$$\sum_{i=1}^{n} w(s_i) = \overleftarrow{w}(\mathcal{B}) = \overrightarrow{w}(\mathcal{B}).$$

Proof. Clearly,

$$\sum_{i=1}^{n} w(s_i) = \frac{\overleftarrow{w}(\mathcal{B}) + \overrightarrow{w}(\mathcal{B})}{2} \ .$$

But by condition (**A**), $\overleftarrow{w}(\mathcal{B}) = \overrightarrow{w}(\mathcal{B})$, hence the thesis. □

Proposition 6.2. **LLL** is stable under cut-elimination.

Proof. Condition (**4**) does not need to be checked, because, as remarked above, we are in a subsystem of **ELL** (modulo the addition of the paragraph modality, which hardly changes anything), and **ELL** is stable under cut-elimination. The other two conditions of Definition 6.3 concern discharged formulas, so we can concentrate on the only two cut-eliminations steps that manipulate them: the additive step and the exponential step. In both cases, we suppose that the proof-net we are reducing satisfies the conditions of Definition 6.3, and we show that the resulting proof-net does too.

In the case of an additive cut, let us call $\mathcal{A}$ the additive box whose principal port is concerned by the cut under reduction. We assume that $\mathcal{A}$ has pad links in its border, otherwise there is nothing to check. Let p be a pad link of $\mathcal{A}$ such that its premise and conclusion have both weight ω, and let l be the why not link (which must exist by the requirement that proof-structures have no discharged formulas among their conclusions) ending the structural branch passing through p; the corresponding premise of l has clearly weight ω. Depending on the component chosen by the reduction, the residue of l either has one premise less, or still has a premise of weight ω. In both cases, no threat of violating the conditions of Definition 6.3 is posed.

Now, let $s_1, \ldots, s_n$ be the conclusions of $\mathcal{A}$ such that $w(s_i) \neq \omega$. For all i such that $0 \leq i \leq n$, let p_i be the pad link whose conclusion is s_i, and let φ_i be the structural branch to which p_i belongs. By condition (c) on boxes in proof-structures (Definition 2.1), if $\mathcal{B}$ is a box (of any kind) containing $\mathcal{A}$ such that one of the φ_j crosses its border, then every φ_i also crosses its border, and if $\mathcal{B}$ is additive, the φ_i all belong to the same component of $\mathcal{B}$. Moreover, by condition (**4**), one of these boxes must be an exponential box, let us call it $\mathcal{C}$.

So let $\mathcal{B}$ be either $\mathcal{C}$ or an additive box contained in $\mathcal{C}$ and containing $\mathcal{A}$. The weight (or left weight, or right weight) of $\mathcal{B}$ is

$$q + \sum_{i=1}^{n} \frac{w(s_i)}{2^k} \ ,$$

where $q \in \mathcal{Q}$ and $k \in \mathbb{N}$ are suitable constants. But by Lemma 6.1,

$$\sum_{i=1}^{n} \frac{w(s_i)}{2^k} = \frac{1}{2^k} \sum_{i=1}^{n} w(s_i) = \frac{1}{2^k} \overleftarrow{w}(\mathcal{A}) = \frac{1}{2^k} \overrightarrow{w}(\mathcal{A}) \ ,$$

so the weight (or left weight, or right weight) of the residue of $\mathcal{B}$ is identical to the weight (or left weight, or right weight) of $\mathcal{B}$, whatever component of $\mathcal{A}$ is chosen. This assures us that conditions (**L**) and (**A**) are preserved.

Let us turn to the case of an exponential cut. Let $\mathcal{B}$ be the !-box whose principal port is concerned by the cut, let l be the why not link also concerned by the cut, and let b be a flat link whose structural branch ends in l. By conditions (**4**) and (**L**), there is exactly one pax link p "between" b and l. Let then x be any link on the directed path from b to p, excluding b itself, let s be the premise of x, and let $\mathcal{C}$ be the box to whose border x belongs. The weight (or left weight, or right weight) of $\mathcal{C}$ is

$$q + w(s) = q + \frac{1}{2^k},$$

where k is the number of links "between" b and x, and $q \in \mathcal{Q}$ is a suitable constant.

Now, after the execution of the cut-elimination step, x is replaced by n copies of a link of the same nature, where $n \geq 1$ is the number of auxiliary ports of $\mathcal{B}$ ($n = 0$ would imply $w(\mathcal{B}) = 0$, violating condition (**L**)), and the residue of $\mathcal{C}$ has weight (or left weight, or right weight)

$$q + \frac{w(\mathcal{B})}{2^k}.$$

But by condition (**L**), $w(\mathcal{B}) = 1$, so the weight (or left weight, or right weight) has not changed, preserving conditions (**L**) and (**A**). □

We shall now prove that the system of Definition 6.3 is indeed Girard's **LLL** without neutrals (which we refer to simply as "Girard's **LLL**"). The proof will use the sequent calculi of both systems, which enable us to reason by induction on the last rule of a proof. On the side of $\mathbf{LL}_\S$, we shall employ the sequent calculus with discharged formulas presented in Sect. 2.3; on the side of Girard's **LLL**, we shall consider the "two-layer" sequent calculus defined in the original paper (Girard, 1998) and recalled in Appendix A.

We first introduce weights in the sequent calculus with discharged formulas, using the notation $\flat A\langle q\rangle$ to denote that an occurrence of discharged formula $\flat A$ has weight q, where q is always an element of $\mathcal{Q}$. If $\Gamma = A_1, \ldots, A_n$, then $\flat\Gamma\langle\vec{q}\rangle$ is short for $\flat A_1\langle q_1\rangle, \ldots, \flat A_n\langle q_n\rangle$. The sequent calculus is defined as the "standard" one, except for additive conjunction, which becomes

$$\frac{\vdash \flat\Delta_1\langle\vec{q_1}\rangle, \Gamma, A \qquad \vdash \flat\Delta_2\langle\vec{q_2}\rangle, \Gamma, B}{\vdash \flat\Delta_1\langle\vec{q_1}/2\rangle, \flat\Delta_2\langle\vec{q_2}/2\rangle, \Gamma, A \mathbin{\&} B}$$

and for the exponential rules, which are replaced by

$$\frac{\vdash \flat\Gamma\langle\vec{q}\rangle, A}{\vdash \flat\Gamma\langle\vec{\omega}\rangle, !A} \qquad \frac{\vdash \Gamma, \flat A\langle q_1\rangle, \ldots, \flat A\langle q_n\rangle}{\vdash \Gamma, ?A} \qquad \frac{\vdash \Gamma, A}{\vdash \Gamma, \flat A\langle 1\rangle} \qquad \frac{\vdash \flat\Gamma\langle\vec{q}\rangle, \Delta}{\vdash \flat\Gamma\langle\vec{\omega}\rangle, \S\Delta}$$

Of course we impose that discharged formulas cannot be principal formulas of any rule except the why not rule.

We call *partial payload* of a sequent $\vdash \flat\Gamma\langle\vec{q}\rangle, \Delta$ the sum of the q_i such that $q_i \neq \omega$, the *full payload* being the sum of *all* q_i. Then, the following is clearly the equivalent of Definition 6.3 for sequent calculus:

Definition 6.4 (LLL, sequent calculus with discharged formulas). LLL is the subsystem of $\mathbf{LL}_\S$ composed of all the derivations $\mathcal{D}$ such that:

(**4**) if $\flat A\langle q\rangle$ is the premise of a why not rule of $\mathcal{D}$, then $q = \omega$;

(**L**) if q is the full payload of the premise of an of course (resp. paragraph) rule of $\mathcal{D}$, then $q = 1$ (resp. $q < \omega$);

(**A**)if q_1, q_2 are the partial payloads of the two premises of a with rule of $\mathcal{D}$, then $q_1 = q_2$.

Notice that condition (**A**) has a very strong consequence on payloads:

Lemma 6.3. The partial payload of any sequent of an **LLL** derivation is an integer.

Proof. The only rule that may introduce non-integer payloads is the with rule, but by condition (**A**), the payload of the conclusion is exactly equal to the payload of the premises (Lemma 6.1). □

Lemma 6.3 justifies the notation $\vdash \flat\Gamma\langle k\rangle, \flat\Delta\langle\omega\rangle, \Sigma$ for the generic **LLL** sequent: $\flat\Gamma$ contains all the discharged formulas whose weight is not ω, so that k is the partial payload of the sequent, $\flat\Delta$ contains all the discharged formulas whose weight is ω, and Σ contains all "plain" formulas of the sequent.

Theorem 6.4 (Soundness of Definition 6.4). For each $\mathbf{LL}_\S$ derivation of the sequent $\vdash \flat\Gamma\langle k\rangle, \flat\Delta\langle\omega\rangle, \Sigma$ respecting the conditions of Definition 6.4, there exist k blocks $\mathbf{A}_1, \dots, \mathbf{A}_k$ such that $\mathbf{A}_1 \uplus \cdots \uplus \mathbf{A}_k = \Gamma$ and a derivation in Girard's **LLL** sequent calculus of the sequent $\vdash \mathbf{A}_1; \dots; \mathbf{A}_k; [\Delta]; \Sigma$.

Proof. The only interesting cases are the additive conjunction and the exponential rules:

With rule. By condition (**A**), our derivation ends with

$$\frac{\vdash \flat\Gamma_1\langle k\rangle, \flat\Delta_1\langle\omega\rangle, \Sigma, A \qquad \vdash \flat\Gamma_2\langle k\rangle, \flat\Delta_2\langle\omega\rangle, \Sigma, B}{\vdash \flat\Gamma\langle k\rangle, \flat\Delta\langle\omega\rangle, \Sigma, A \mathbin{\&} B}$$

where $\Gamma = \Gamma_1 \uplus \Gamma_2$ and $\Delta = \Delta_1 \uplus \Delta_2$. By induction hypothesis, we have a proof of $\vdash \mathbf{A}^1_1; \dots; \mathbf{A}^1_k; [\Delta_1]; \Sigma; A$ and a proof of $\vdash \mathbf{A}^2_1; \dots; \mathbf{A}^2_k; [\Delta_2]; \Sigma; B$. If we put $\mathbf{A}_i = \mathbf{A}^1_i \uplus \mathbf{A}^2_i$ and $\Delta = \Delta_1 \uplus \Delta_2$, using additive and multiplicative weakenings we obtain

$$\frac{\dfrac{\vdash \mathbf{A}^1_1; \dots; \mathbf{A}^1_k; [\Delta_1]; \Sigma; A}{\vdash \mathbf{A}_1; \dots; \mathbf{A}_k; [\Delta]; \Sigma; A} \qquad \dfrac{\vdash \mathbf{A}^2_1; \dots; \mathbf{A}^2_k; [\Delta_2]; \Sigma; A}{\vdash \mathbf{A}_1; \dots; \mathbf{A}_k; [\Delta]; \Sigma; A}}{\vdash \mathbf{A}_1; \dots; \mathbf{A}_k; [\Delta]; \Sigma; A \mathbin{\&} B}$$

which clearly verifies $\mathbf{A}_1 \uplus \cdots \uplus \mathbf{A}_k = \Gamma$.

Of course rule. By condition (**L**), our derivation ends with

$$\frac{\vdash \flat\Gamma\langle 1\rangle, A}{\vdash \flat\Gamma\langle\omega\rangle, !A}$$

Using the induction hypothesis, if $\Gamma = B_1, \dots, B_n$, we have a proof of $\vdash B_1, \dots, B_n; A$ in Girard's sequent calculus, from which we obtain $\vdash [B_1]; \dots; [B_n]; !A$ by means of an of course rule.

Why not rule. By condition (**4**), our derivation ends with

$$\frac{\vdash \flat\Gamma\langle k\rangle, \flat\Delta\langle\omega\rangle, \Sigma, \flat A\langle\omega\rangle, \dots, \flat A\langle\omega\rangle}{\vdash \flat\Gamma\langle k\rangle, \flat\Delta\langle\omega\rangle, \Sigma, ?A}$$

which means that by induction hypothesis we have a proof of

$\vdash \mathbf{A}_1; \ldots; \mathbf{A}_k; [\Delta]; \Sigma; [A]; \ldots; [A]$ in Girard's sequent calculus. By repeated application of multiplicative contractions and a why not rule, we obtain

$$\dfrac{\dfrac{\vdash \mathbf{A}_1; \ldots; \mathbf{A}_k; [\Delta]; \Sigma; [A]; \ldots; [A]}{\vdash \mathbf{A}_1; \ldots; \mathbf{A}_k; [\Delta]; \Sigma; [A]}}{\vdash \mathbf{A}_1; \ldots; \mathbf{A}_k; [\Delta]; \Sigma; ?A}$$

Flat rule. Our derivation ends with

$$\dfrac{\vdash \flat\Gamma\langle k\rangle, \flat\Delta\langle\omega\rangle, \Sigma, A}{\vdash \flat\Gamma\langle k\rangle, \flat\Delta\langle\omega\rangle, \Sigma, \flat A\langle 1\rangle}$$

By induction hypothesis, we have a proof of $\vdash \mathbf{A}_1; \ldots; \mathbf{A}_k; [\Delta]; \Sigma; A$. But this proof is already what we are looking for, since A is itself a block, and $\mathbf{A}_1 \uplus \cdots \uplus \mathbf{A}_k \uplus \{A\} = \Gamma \uplus \{A\}$.

Paragraph rule. By condition (**L**), our derivation ends with

$$\dfrac{\vdash \flat\Gamma\langle k\rangle, \Sigma}{\vdash \flat\Gamma\langle\omega\rangle, \S\Sigma}$$

The induction hypothesis gives us a proof of $\vdash \mathbf{A}_1; \ldots; \mathbf{A}_k; \Sigma$, where the additive block are such that $\mathbf{A}_1 \uplus \cdots \uplus \mathbf{A}_k = \Gamma$. From this, after applying a paragraph rule we obtain $\vdash [\Gamma]; \S\Sigma$, which is what we want.

□

As we have seen in the above proof, any sequent of Girard's calculus for **LLL** can be written as $\vdash \mathbf{A}_1; \ldots; \mathbf{A}_k; [\Delta]; \Sigma$. In what follows, we make the additional requirement that the $\mathbf{A}_i$ are "real" additive blocks, i.e., they contain at least two undischarged formulas; blocks containing only one discharged or undischarged formula are supposed to be resp. in $[\Delta]$ and Σ.

Let us introduce some useful notations:

— we write $\Gamma' \sqsubseteq \Gamma$ iff $\Gamma = A_1, \ldots, A_n$ and there exist n non-negative integers $m_1, \ldots, m_n$ such that $\Gamma' = A_1^{(m_1)}, \ldots, A_n^{(m_n)}$, where $A_i^{(m_i)}$ stands for m_i occurrences of A_i;
— we write $\Gamma' \sqsubseteq^* \Gamma$ iff $\Gamma' \sqsubseteq \Gamma$ and $\Gamma' \neq \emptyset$;
— $\flat\Gamma\langle q\rangle$ is short for $\flat\Gamma\langle\vec{q}\rangle$ such that $\sum_i q_i = q$.

Lemma 6.5.

1 *Union*: if $\Gamma'_1, \Gamma'_2 \sqsubseteq \Gamma$ then $\Gamma'_1 \uplus \Gamma'_2 \sqsubseteq \Gamma$.
2 *Weakening*: if $\Gamma' \sqsubseteq \Gamma$, then, for any Δ, $\Gamma' \sqsubseteq \Gamma \uplus \Delta$.
3 *Contraction*: if $\Gamma' \sqsubseteq \Gamma \uplus \{A, A\}$, then $\Gamma' \sqsubseteq \Gamma \uplus \{A\}$.
4 *Non-empty case*: all of the above hold replacing $\sqsubseteq$ with $\sqsubseteq^*$.

Proof. Obvious. □

Theorem 6.6 (Completeness of Definition 6.4). For each derivation of $\vdash \mathbf{A}_1; \ldots; \mathbf{A}_k; [\Delta]; \Sigma$ in Girard's sequent calculus for **LLL**, there exists an $\mathbf{LL}_\S$ derivation $\mathcal{D}$ of $\vdash \flat\Gamma_1\langle 1\rangle, \ldots, \flat\Gamma_k\langle 1\rangle, \flat\Delta'\langle\omega\rangle, \Sigma$ such that:

— $\mathcal{D}$ satisfies the conditions of Definition 6.4;

— for all i such that $1 \leq i \leq k$, $\Gamma_i \sqsubseteq^* \mathbf{A}_i$;
— $\Delta' \sqsubseteq \Delta$.

Proof. Identity, cut, multiplicative, quantifier, and additive disjunction rules are simulated by their direct counterparts without problems. For what concern the other rules, we have:

With rule. Our derivation ends with

$$\frac{\vdash \mathbf{A}_1;\ldots;\mathbf{A}_k;[\Delta];\Sigma;A \qquad \vdash \mathbf{A}_1;\ldots;\mathbf{A}_k;[\Delta];\Sigma;B}{\vdash \mathbf{A}_1;\ldots;\mathbf{A}_k;[\Delta];\Sigma;A \,\&\, B}$$

By induction hypothesis, we can build the following proof:

$$\frac{\vdash \flat\Gamma_1^1\langle 1\rangle;\ldots;\flat\Gamma_k^1\langle 1\rangle;\flat\Delta_1'\langle\omega\rangle;\Sigma;A \qquad \vdash \flat\Gamma_1^2\langle 1\rangle;\ldots;\flat\Gamma_k^2\langle 1\rangle;\flat\Delta_2'\langle\omega\rangle;\Sigma;B}{\vdash \flat\Gamma_1^1\langle 1/2\rangle,\flat\Gamma_1^2\langle 1/2\rangle,\ldots,\flat\Gamma_k^1\langle 1/2\rangle,\flat\Gamma_k^2\langle 1/2\rangle,\flat\Delta_1'\langle\omega\rangle,\flat\Delta_2'\langle\omega\rangle,\Sigma,A \,\&\, B}$$

Now, if we pose $\Gamma_i = \Gamma_i^1 \uplus \Gamma_i^2$ and $\Delta' = \Delta_1' \uplus \Delta_2'$, we have that the overall weight of each $\flat\Gamma_i$ is 1, and by Lemma 6.5 we have $\Gamma_i \sqsubseteq^* \mathbf{A}_i$ and $\Delta' \sqsubseteq \Delta$.

Of course rule. Our derivation ends with

$$\frac{\vdash B_1,\ldots,B_n;A}{\vdash [B_1];\ldots;[B_n];!A}$$

Applying the induction hypothesis, we obtain a proof of $\vdash \flat\Gamma\langle 1\rangle, A$, where $\Gamma = B_1^{(m_1)},\ldots,B_n^{(m_n)}$ and at least one m_i is non-null. Then, we are in position to apply an of course rule respecting condition (**L**), to obtain $\vdash \flat\Gamma\langle\omega\rangle, !A$, and $\Gamma \sqsubseteq \{B_1,\ldots,B_n\}$ already holds by induction.

Why not rule. Our derivation ends with

$$\frac{\vdash \mathbf{A}_1;\ldots;\mathbf{A}_k;[\Delta];\Sigma;[A]}{\vdash \mathbf{A}_1;\ldots;\mathbf{A}_k;[\Delta];\Sigma;?A}$$

By induction hypothesis, we have a proof of the sequent

$$\vdash \flat\Gamma_1\langle 1\rangle,\ldots,\flat\Gamma_k\langle 1\rangle,\flat\Delta'\langle\omega\rangle,\Sigma,\flat\Delta''\langle\omega\rangle$$

where $\Gamma_i \sqsubseteq^* \mathbf{A}_i$, $\Delta' \sqsubseteq \Delta$, and most importantly $\Delta'' \sqsubseteq \{A\}$. This means that, if we put $\Phi = \flat\Gamma_1\langle 1\rangle,\ldots,\flat\Gamma_k\langle 1\rangle,\flat\Delta'\langle\omega\rangle,\Sigma$, our sequent actually looks like $\vdash \Phi,\flat A\langle\omega\rangle,\ldots,\flat A\langle\omega\rangle$, so we can apply a why not rule respecting condition (**4**), obtaining $\vdash \Phi, ?A$, which is exactly what we are looking for.

Paragraph rule. This case is virtually identical to that of the of course rule.

Structural rules. Let $\vdash \mathbf{A}_1;\ldots;\mathbf{A}_k;[\Delta];\Sigma$ be the premise of our last rule. In the case of a multiplicative (resp. additive) weakening, we have that Δ becomes $\Delta \uplus \{A\}$ (resp. some $\mathbf{A}_i$ becomes $\mathbf{A}_i \uplus \{A\}$). By the induction hypothesis, we have a proof ending with $\vdash \flat\Gamma_1\langle 1\rangle,\ldots,\flat\Gamma_k\langle 1\rangle,\flat\Delta'\langle\omega\rangle,\Sigma$ such that $\Delta' \sqsubseteq \Delta$ (resp. $\Gamma_i \sqsubseteq^* \mathbf{A}_i$). But then, by Lemma 6.5, $\Delta' \sqsubseteq \Delta \uplus \{A\}$ (resp. $\Gamma_i \sqsubseteq^* \mathbf{A}_i \uplus \{A\}$), so we do not need to do anything. In the case of a multiplicative (resp. additive) contraction, we have that $\Delta = \Delta'' \uplus \{A, A\}$ becomes $\Delta'' \uplus \{A\}$ (resp. for some i, $\mathbf{A}_i = \mathbf{A}_i' \uplus \{A, A\}$ becomes $\mathbf{A}_i' \uplus \{A\}$). Again, by Lemma 6.5, we still have $\Delta' \sqsubseteq \Delta'' \uplus \{A\}$ (resp. $\Gamma_i \sqsubseteq^* \mathbf{A}_i' \uplus \{A\}$), so we have nothing to do.

□

Combined together, the two theorems above prove that we have actually captured Girard's **LLL**: if $A_1, \ldots, A_n$ are "plain" $\mathbf{LL}_\S$ formulas, $\vdash A_1; \ldots; A_n$ is provable in Girard's **LLL** iff $\vdash A_1, \ldots, A_n$ is provable in $\mathbf{LL}_\S$ with a derivation satisfying Definition 6.4. Moreover, there is a tight correspondence (which is not one-to-one only because of minor structural details) between the proofs of the two systems.

This correspondence gives a nice insight on the nature of **LLL** blocks: they are lists of discharged formulas of overall weight 1; this is why they "count" as one discharged formula.

6.2. *Soft Linear Logic*

While we are at it, let us also give a reformulation of Lafont's **SLL** (Lafont, 2004) in terms of weighed proof-nets and sequent calculus with discharged formulas. We remind that **SLL** is obtained by replacing the four exponential rules of **LL** sequent calculus (Sect. 2.3) with

$$\frac{\vdash \Gamma, A}{\vdash ?\Gamma, !A}\ \text{(soft promotion)} \qquad\qquad \frac{\vdash \Gamma, A, \ldots, A}{\vdash \Gamma, ?A}\ \text{(multiplexing)}$$

Definition 6.5 (SLL, weighed proof-nets). **SLL** is the subsystem of **LL** composed of all the proof-nets π such that:

(**T**) if $\mathcal{B}$ is a !-box of π, then $w(\mathcal{B}) < \omega$;
(**S**) if $w(s) = \omega$, then s is the premise of a unary *why not* link.
(**A′**) if $\mathcal{B}$ is an additive box of π, then $\overleftarrow{w}(\mathcal{B}) = \overrightarrow{w}(\mathcal{B}) = 0$.

As we have already pointed out, condition (**T**) forbids digging; condition (**S**) forces soft promotion, and the additional condition (**A′**) simply states that additive boxes in **SLL** cannot have any *pad* link.

The following result is actually much simpler to prove than its **LLL** counterpart above, so the proof is omitted:

Theorem 6.7. For each **SLL** derivation of $\vdash \Gamma$ there is an **LL** proof-net of conclusions Γ satisfying Definition 6.5, and vice-versa.

7. Conclusions

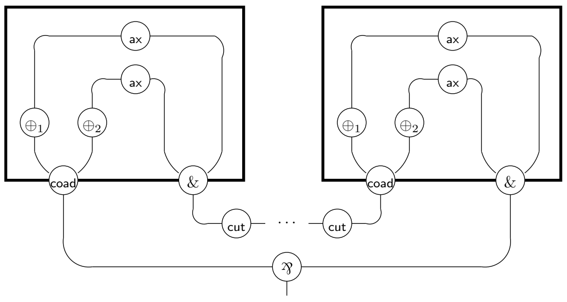
We have seen how Danos and Joinet's approach can be extended to obtain a purely logical characterization of **FP**, arguably simpler than **LLL**. We have also enhanced Danos and Joinet's ideas in order to recover Girard's **LLL** without changing the syntax of linear logic (except for the addition of the paragraph modality), but rather by imposing structural constraints on proofs. We would like to make here a few concluding remarks.

Lazy cut-elimination and polytime strong normalization. Our work excludes *a priori* the reduction of additive commutative cuts, i.e., cuts whose one of the premises is the conclusion of an additive contraction link. In Sect. 2 we said that the reason for ignoring these cuts is that they pose non-confluence problems; we have seen in the completeness proof that this restriction is not harmful, since Theorem 3.3 applies to proof-nets representing data and functions on integers or binary strings, and we are sure that the cut-elimination process ends with a truly cut-free proof-net even without reducing additive commutative cuts.

There is actually another reason justifying the use of lazy cut-elimination: without it, strong polytime normalization for $\overline{\mathbf{LLL}}$ would fail. Indeed, consider the additive η-expansion of the identity, which corresponds to the λ-term

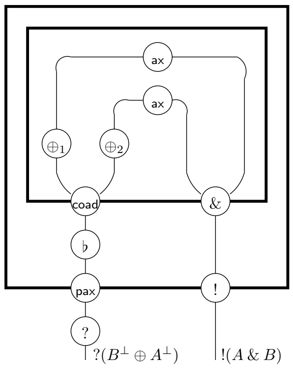
$$A = \lambda x.\langle \mathsf{fst}\ x, \mathsf{snd}\ x\rangle\ .$$

If we apply the nth Church integer to A, we obtain a term for which, if we admit additive commutative reductions, there exists a reduction sequence of length $O(2^n)$. In terms of proof-nets, after a number of steps equal to n plus a constant, the reduction stumbles upon the following "chain" composed of n additive boxes (we omit the types, which are irrelevant):



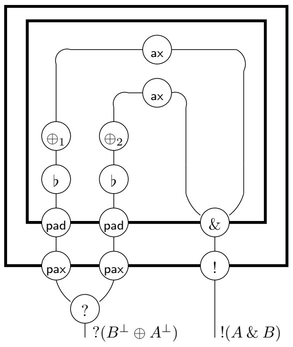
Lazy cut-elimination would stop at this moment; on the contrary, allowing the reduction of additive commutative cuts duplicates the boxes, and if no reduction inside an additive box is performed before all reductions outside the boxes are completed, we get an exponentially big proof-net.

In **LLL** (we mean the system introduced in Definition 6.3) the situation is even worse: strong polytime normalization fails even under lazy cut-elimination. As a matter of fact, we can consider a construction similar to the one above, but instead of η-expanding the identity on $A \& B$, we η-expand the identity on $!(A \& B)$, obtaining the proof-net



(only the conclusions are given, the other types can be uniquely inferred from them).

Now we "turn" the additive contraction into a multiplicative one, and build the following proof-net, which we call ρ:

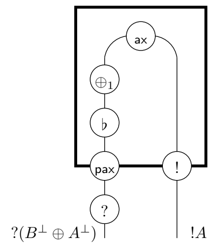


ρ is clearly not in $\overline{\mathbf{LLL}}$ (it violates condition **i** of Definition 3.4), but the reader can check that it satisfies the requirements of Definition 6.3, thus being a valid **LLL** proof-net. In fact, passing from the η-expansion of $!(A \,\&\, B) \multimap\, !(A \,\&\, B)$ to ρ is possible thanks to the exponential isomorphism, in particular the direction which holds in **LLL** but is false in $\overline{\mathbf{LLL}}$, namely $!A \otimes\, !B \multimap\, !(A \,\&\, B)$. In categorical terms, the fact that an additive contraction can be turned into a multiplicative one is justified by the commutation of the following diagram:

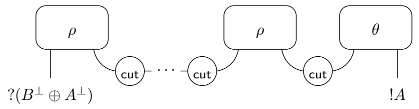
$$\begin{tikzcd}
!C \arrow[d, "ctr"'] \arrow[rrd, "!\langle f{,}g\rangle"] & & \\
!C \otimes !C \arrow[r, "!f \otimes !g"'] & !A \otimes !B \arrow[r, "iso"'] & !(A \,\&\, B)
\end{tikzcd}$$

where f, g are two morphisms of source C and resp. targets A and B, $\langle \cdot, \cdot \rangle$ denotes the cartesian product of two morphisms, $!(\cdot)$ and $\cdot \otimes \cdot$ are resp. the bang and tensor functors, and *ctr* and *iso* are resp. the contraction map on $!C$ and the morphism from $!A \otimes\, !B$ to $!(A \,\&\, B)$ which represents one side of the exponential isomorphism. Clearly $!\langle f, g\rangle$ uses additive contraction, while $iso \circ\, !f \otimes !g \circ ctr$ does not (it uses multiplicative contraction instead).

If we define θ to be the proof-net (which is also in **LLL**)



we can build the following "chain" of length n, which we call χ_n:



Now, χ_n has size linear in n, and moreover it is a lazy proof-net (we can assume A and B to be propositional variables). Therefore, it can be brought to its cut-free form without reducing additive commutative cuts; yet, if we always reduce cuts at depth zero first, we obtain a reduction sequence of length $O(2^n)$.

Girard (Girard, 1998) avoids this problem by reducing exponential cuts in a careful way, taking into account some conditions on his boolean weights. In our setting, his conditions can be reformulated as follows. Suppose that a !-box $\mathcal{B}$ is cut with an n-ary why not link l. The cut-elimination procedure we have considered so far reduces this cut by finding the flat links "above" l and dispatching one copy of the contents of $\mathcal{B}$ to each of these n links. Here, we dispatch a copy of the contents of $\mathcal{B}$ to a flat link b only if the exponential branch starting from b and ending into l does not cross any pad link. The arity of l is thus reduced, but maybe not enough to make it disappear, in which case a residue of $\mathcal{B}$ and of l can still be found in the reduct, still cut one with the other, until some additive reduction "unlocks" the situation.

The reader may check that this "hyper-lazy" procedure is linear in the case of χ_n, no matter what reduction sequence is chosen. We believe that this is not an accident, and indeed we conjecture that **LLL** is polytime strongly normalizing under this cut-elimination procedure (of course for lazy proof-nets only).

Following a remark of Terui (Terui, 2002), this would seem to exclude the possibility of encoding Bellantoni & Cook's safe recursion (Bellantoni and Cook, 1992) in light linear logic even with the aid of addive connectives.

The paragraph modality. Imagine to take an $\overline{\mathbf{LLL}}/\mathbf{LLL}$ proof-net and "strip off" all of its §-boxes, erasing at the same time every § from its formulas. It is not hard to see that what we get is a correct **LL** proof-net, which almost certainly violates the stratification condition, but whose normalization obviously takes no more steps than the original $\overline{\mathbf{LLL}}/\mathbf{LLL}$ proof-net (there are fewer cuts to reduce!).

This means that §-boxes do not play any role in the dynamics of cut-elimination, in particular with respect to its efficiency. Our work confirms the idea, already suggested by Asperti and Roversi (Asperti and Roversi, 2002), that the fundamental nature of the § modality is that of a statical *depth-marker*.

In light logics, we need to be able to delimit sub-proof-nets which can communicate only with a limited number of other sub-proof-nets, namely those at the same depth; one could imagine to do this by "framing" **LL** sub-proof-nets in order to restrict their interactions. The § modality internalizes this restriction into the logic: a "framed" A should not interact with $A^\perp$, so we turn it into $\S A$; we then see how our "frames" become §-boxes.

This casts doubts on the logical value of the paragraph modality, and may justify why it has not so far received any interesting semantical interpretation.

References

Asperti, A. and Roversi, L. (2002). Intuitionistic Light Affine Logic. *ACM Trans. Comput. Log.*, 3(1):137–175.

Baillot, P. (2004a). Stratified coherent spaces: a denotational semantics for Light Linear Logic. *Theoret. Comput. Sci.*, 318(1–2):29–55.

Baillot, P. (2004b). Type Inference for Light Affine Logic via Constraints on Words. *Theoret. Comput. Sci.*, 328(3):289–323.

Bellantoni, S. and Cook, S. (1992). A new recursion-theoretic characterization of the polytime functions. *Computational Complexity*, 2:97–110.

Coppola, P. and Martini, S. (2001). Typing λ-terms in Elementary Logic with Linear Constraints. In *Proceedings of TLCA'01*, pages 76–90. LNCS 2044, Springer-Verlag.

Danos, V. and Joinet, J.-B. (2003). Linear logic & elementary time. *Inform. and Comput.*, 183:123–137.

Girard, J.-Y. (1996). Proof-nets: The parallel syntax for proof-theory. In Ursini and Agliano, editors, *Logic and Algebra.* Marcel Dekker, Inc.

Girard, J.-Y. (1998). Light linear logic. *Inform. and Comput.*, 14(3):175–204.

Kanovich, M., Okada, M., and Scedrov, A. (2003). Phase semantics for light linear logic. *Theoret. Comput. Sci.*, 294(3):525–549.

Lafont, Y. (2004). Soft linear logic and polynomial time. *Theoret. Comput. Sci.*, 318(1–2):163–180.

Laurent, O. and de Falco, L. T. (2004). Slicing polarized additive normalization. In Ehrhard, T., Girard, J.-Y., Ruet, P., and Scott, P., editors, *Linear Logic in Computer Science*, Lecture Notes Series 316, pages 247–282. London Mathematical Society.

Laurent, O. and de Falco, L. T. (2005). Obsessional cliques: a semantic characterization of bounded time complexity. *In preparation.*

Mairson, H. and Terui, K. (2003). On the computational complexity of cut-elimination in linear logic. In *Proceedings of ICTCS 2003*, volume 2841 of *LNCS*, pages 23–36. Springer-Verlag.

Roversi, L. (1999). A P-time completeness proof for light logics. In *Proceedings of CSL'99*, volume 1683 of *LNCS*, pages 469–483. Springer-Verlag.

Terui, K. (2002). Light affine lambda calculus and polynomial time strong normalization. To appear in *Arch. Math. Logic.* Extended abstract appeared in the Proceedings of LICS 2001, IEEE Computer Society, pp. 209–220.

Terui, K. (2004). Light affine set theory: a naive set theory of polynomial time. *Studia Logica*, 77:9–40.

Tortora de Falco, L. (2003). Additives of linear logic and normalization – Part I: a (restricted) Church-Rosser property. *Theoret. Comput. Sci.*, 294(3):489–524.

Appendix A. LLL sequent calculus

The material in this appendix is taken from (Girard, 1998).

We briefly recall the definition of Girard's **LLL** sequents:

- ▶ A *discharged formula* is an expression $[A]$, where A is a formula;
- ▶ a *block* $\mathbf{A}$ is a sequence $A_1, \dots, A_n$ of formulas, or a single discharged formula $[A]$;
- ▶ a *sequent* is an expression $\vdash \mathbf{A}_1, \dots, \mathbf{A}_n$, where $\mathbf{A}_1, \dots, \mathbf{A}_n$ are blocks.

The intuitive meaning of a block $A_1, \dots, A_n$ is $A_1 \oplus \cdots \oplus A_n$; the intuitive meaning of a discharged formula $[A]$ is $?A$; if $\mathbf{A}_1, \dots, \mathbf{A}_n$ correspond to the formulas $A_1, \dots, A_n$, then the intuitive meaning of the sequent $\vdash \mathbf{A}_1, \dots, \mathbf{A}_n$ is $A_1 \parr \cdots \parr A_n$.

Identity/Negation

$$\frac{}{\vdash A; A^{\perp}}\ (\text{identity}) \qquad \frac{\vdash \Gamma; A \qquad \vdash \Delta; A^{\perp}}{\vdash \Gamma; \Delta}\ (\text{cut})$$

Structure

$$\frac{\vdash \Gamma; \mathbf{A}; \mathbf{B}; \Delta}{\vdash \Gamma; \mathbf{B}; \mathbf{A}; \Delta}\ (\text{M-exchange}) \qquad \frac{\vdash \Gamma; \mathbf{A}, C, D, \mathbf{B}}{\vdash \Gamma; \mathbf{A}, D, C, \mathbf{B}}\ (\text{A-exchange})$$

$$\frac{\vdash \Gamma}{\vdash \Gamma; [A]}\ (\text{M-weakening}) \qquad \frac{\vdash \Gamma; \mathbf{A}}{\vdash \Gamma; \mathbf{A}, B}\ (\text{A-weakening})$$

$$\frac{\vdash \Gamma; [A]; [A]}{\vdash \Gamma; [A]}\ (\text{M-contraction}) \qquad \frac{\vdash \Gamma; \mathbf{A}, B, B}{\vdash \Gamma; \mathbf{A}, B}\ (\text{A-contraction})$$

Logic

$$\frac{\vdash \Gamma; A \qquad \vdash B; \Delta}{\vdash \Gamma; A \otimes B; \Delta}\ (\text{times}) \qquad \frac{\vdash \Gamma; A; B}{\vdash \Gamma; A \mathbin{⅋} B}\ (\text{par})$$

$$\frac{\vdash \Gamma; A \qquad \vdash \Gamma; B}{\vdash \Gamma; A \,\&\, B}\ (\text{with}) \qquad \frac{\vdash \Gamma; A_i}{\vdash \Gamma; A_1 \oplus A_2}\ (\text{plus } i) \quad i \in \{1, 2\}$$

$$\frac{\vdash B_1, \ldots, B_n; A}{\vdash [B_1]; \ldots; [B_n]; !A}\ (\text{of course}) \qquad \frac{\vdash \Gamma; [A]}{\vdash \Gamma; ?A}\ (\text{why not})$$

$$\frac{\vdash B_1 | \ldots | B_n; A_1; \ldots; A_m}{\vdash [B_1]; \ldots; [B_n]; \S A_1; \ldots; \S A_m}\ (\text{neutral}) \quad \text{`$|$' stands for `,' or `;'}$$

$$\frac{\vdash \Gamma; A}{\vdash \Gamma; \forall X A}\ (\text{for all}) \quad X \text{ not free in } \Gamma \qquad \frac{\vdash \Gamma; A[B/X]}{\vdash \Gamma; \exists X A}\ (\text{there is})$$